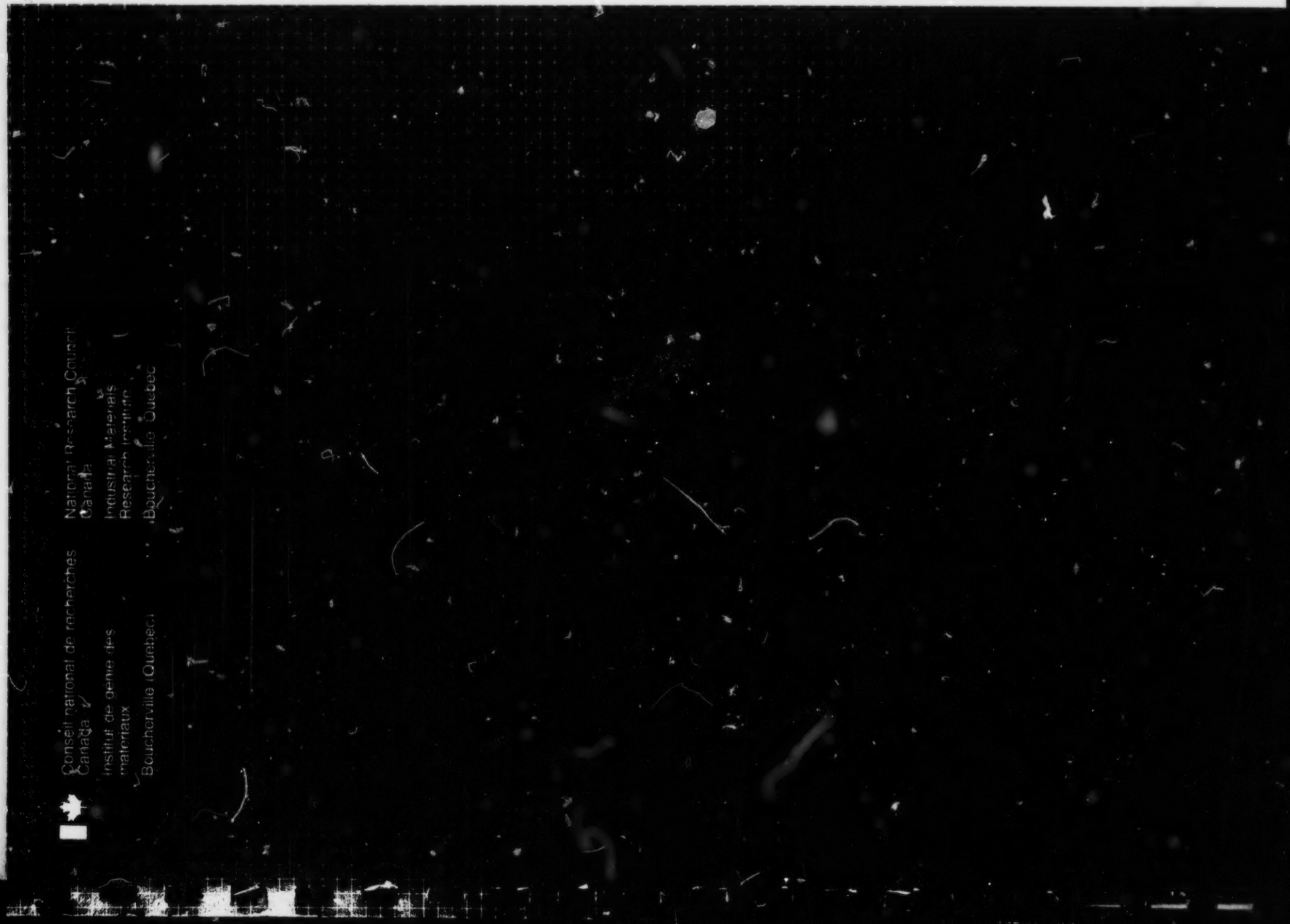
# ASSERVISSEMENT DES PARAMETRES EN SOUDAGE A L'AIDE D'UN SYSTEME DE VISION PASSIVE (PHASE 2)



## REMERCIEMENTS

Nous remercions M. Jean-Guy Allard pour sa collaboration précieuse au projet.

IGM86-201-402
CNRC 26510

# ASSERVISSEMENT DES PARAMETRES EN SOUDAGE A L'AIDE D'UN SYSTEME DE VISION PASSIVE (PHASE 2)

CONTRAT NO OSD85-00161

RC-285

Directeur du projet: Pierre Fafard

Auteurs du rapport: Yves Chartier
Pierre Fafard

Collaborateurs: Serge Landry
Michel Motard
Guy Patenaude
Jean Tremblay

Approuvé par:

Dr. N. F. Eaton

# I. TABLE DES MATIERES


| | | page |
|---|---|---|
| I. | TABLE DES MATIERES ..... | i |
| II. | LISTE DES FIGURES ..... | ii |
| III. | LISTE DES TABLEAUX ..... | v |
| 1. | INTRODUCTION ..... | 1 |
| 2. | INTEGRATION DU MATERIEL ..... | 5 |
| 2.1 | SYSTEME OPTIQUE ..... | 5 |
| 3. | LOGICIEL DU SYSTEME DE VISION INFRA-ROUGE ..... | 10 |
| 3.1 | INTRODUCTION ..... | 10 |
| 3.2 | STRUCTURE DU PROGRAMME *SUIVI.C* ..... | 11 |
| 3.3 | ALGORITHME DE RECHERCHE DU JOINT ..... | 13 |
| 4. | ESSAIS ET INTERPRETATION DES RESULTATS ..... | 14 |
| 4.1 | ESSAIS SUR UN JOINT BOUT A BOUT AVEC PREPARATION EN V (PLEINE IMAGE) ..... | 19 |
| 4.2 | ESSAIS SUR UN JOINT BOUT A BOUT A BORD DROIT (PLEINE IMAGE) ..... | 40 |
| 4.3 | ESSAIS SUR UN JOINT BOUT A BOUT AVEC PREPARATION EN V (UTILISATION D'UNE FENTE) ..... | 46 |
| 4.4 | ESSAIS SUR UN JOINT BOUT A BOUT A BORD DROIT (UTILISATION D'UNE FENTE) ..... | 55 |
| 4.5 | ESSAIS SUR UN JOINT A RECOUVREMENT ET SUR UN JOINT D'ANGLE (PLEINE IMAGE) ..... | 64 |
| 4.6 | ESSAIS DE SUIVI DE JOINT ..... | 67 |
| 5. | CONCLUSION ..... | 73 |
| ANNEXE A | Logiciel PSEUDO.C | |
| ANNEXE B | Logiciel SUIVI.C | |
| ANNEXE C | Logiciel SUITE.C | |

## II. LISTE DES FIGURES


| | | page |
|---|---|---|
| 2.1 | Système optique et caméra AGA ..... | 7 |
| 2.2 | Spectre IR moyen (transmission) du kapton ..... | 8 |
| 2.3 | Installation de la caméra AGA thermovision ..... | 9 |
| 4.0 | Dimensions d'un bain de fusion ..... | 15 |
| 4.1 | Echantillon d'une image ..... | 20 |
| 4.2 | Essai 000 (joint bout à bout avec préparation en V) ... | 23 |
| 4.3 | Essai 001 (joint bout à bout avec préparation en V) ... | 24 |
| 4.4 | Essai 002 (joint bout à bout avec préparation en V) ... | 25 |
| 4.5 | Essai 003 (joint bout à bout avec préparation en V) ... | 26 |
| 4.6 | Essai 004 (joint bout à bout avec préparation en V) ... | 27 |
| 4.7 | Essai 005 (joint bout à bout avec préparation en V) ... | 28 |
| 4.8 | Essai 008 (joint bout à bout avec préparation en V) ... | 29 |
| 4.9 | Essai 011 (joint bout à bout avec préparation en V) ... | 30 |
| 4.10 | Essai 012 (joint bout à bout avec préparation en V) ... | 31 |
| 4.11 | Essai 014 (joint bout à bout avec préparation en V) ... | 32 |
| 4.12 | Essai 009 (joint bout à bout avec préparation en V) ... | 36 |
| 4.13 | Essai 11A (joint bout à bout avec préparation en V) ... | 47 |

page

4.14 Essai 11B (joint bout à bout avec préparation en V) ... 48

4.15 Essai 11C (joint bout à bout avec préparation en V) ... 49

4.16 Essai 11D (joint bout à bout avec préparation en V) ... 50

4.17 Essai 11E (joint bout à bout avec préparation en V) ... 51

4.18 Essai 11F (joint bout à bout avec préparation en V) ... 52

4.19 Essai 11G (joint bout à bout avec préparation en V) ... 53

4.20 Essai 11H (joint bout à bout avec préparation en V) ... 54

4.21 Essai 14 (joint bout à bout avec préparation en V) .... 37

4.22 Essai 14A (joint bout à bout avec préparation en V) ... 38

4.23 Essai 14B (joint bout à bout avec préparation en V) ... 33

4.24 Essai 14C (joint bout à bout avec préparation en V) ... 34

4.25 Essai 14D (joint bout à bout avec préparation en V) ... 39

4.26 Essai 017 (joint bout à bout à bords droits) ..... 56

4.27 Essai 018 (joint bout à bout à bords droits) ..... 57

4.28 Essai 019 (joint bout à bout à bords droits) ..... 58

4.29 Essai 020 (joint bout à bout à bords droits) ..... 59

4.30 Essai 021 (joint bout à bout à bords droits) ..... 60

4.31 Essai 022 (joint bout à bout à bords droits) ..... 61

page

4.32 Essai 023 (joint bout à bout à bords droits) ..... 62

4.33 Essai 024 (joint bout à bout à bords droits) ..... 63

4.34 Essai 047 (joint bout à bout à bords droits) ..... 41

4.35 Essai 049 (joint bout à bout à bords droits) ..... 42

4.36 Essai 051 (joint bout à bout à bords droits) ..... 43

4.37 Essai 053 (joint bout à bout à bords droits) ..... 44

4.38 Essai 054 (joint bout à bout à bords droits) ..... 45

4.39 Essai 043 (joint bout à bout à angle) ..... 65

4.40 Essai 045 (joint bout à bout à recouvrement) ..... 66

4.41 Profils pour un joint bout à bout à bord droit ..... 68

4.42 Profil pour un joint bout à bout avec préparation en V . . 70

4.43 Coupon d'essai, plaque droite avec préparation en V et inclinaison de cinq degrés ..... 71

4.44 Coupon d'essai, plaque curviligne avec préparation à bords droits ..... 72

## III. LISTE DES TABLEAUX


| | | page |
|---|---|---|
| 1 | Essais sur joints bout à bout avec préparation en V .... | 16 |
| 2 | Essais sur joints bout à bout à bords droits ..... | 17 |

# 1. INTRODUCTION

En 1980, le Centre du Québec de l'Institut de Soudage du Canada (ISC) et le Conseil national de recherches du Canada, division Institut de génie des matériaux (IGM), décidèrent d'entreprendre des recherches dans le domaine de l'automatisation des procédés de soudage à l'arc électrique.

Il apparût alors deux options possibles. La première consistait à développer des mécanismes permettant d'augmenter la précision de la fabrication et le positionnement des pièces à souder. La deuxième option impliquait de modifier en temps réel, à partir d'information provenant de différents capteurs , les paramètres nécessaires à l'obtention d'une soudure de qualité. En d'autres termes, il s'agissait de réaliser un système adaptatif permettant le suivi du joint ainsi que le contrôle des bilans matière et énergie de l'opération de soudage.

La première option demandant des équipements mécaniques de positionnement de haute précision, c'est-à-dire des coûts très élevés, fût donc rejetée. Du point de vue des coûts seulement, le développement de systèmes adaptatifs devenait donc justifiable. L'approche originale qui fût prise par l'ISC et l'IGM consistait à analyser l'information provenant du rayonnement infra-rouge des plaques à souder. Un premier projet (1) démontra la faisabilité d'une telle approche. Par la suite dans le cadre de projets subséquents (2-3-4-5), un suiveur de joint optique infra-rouge fut développé et montra hors de tout doute le potentiel énorme de la technologie.

(1)- Contrat CNRC No. OSD80-00034
(2)- Contrat CNRC No. OSD81-00057
(3)- Contrat CNRC No. OSD82-00042
(4)- Contrat CNRC No. OSD81-00041
(5)- Contrat CNRC No. OSD83-00035

L'objet du présent projet concerne donc la réalisation et l'implantation d'un suiveur de joint infra-rouge de troisième génération pour le soudage à l'arc électrique. Le suiveur de la première génération utilisait un système de détection avec balayage mécanique. Les limites relatives à la courbure des joints provenaient essentiellement de la faible vitesse d'acquisition des données due au balayage mécanique.

Une autre limite du suiveur de première génération tenait au fait que le balayage était toujours perpendiculaire à la direction d'avance de la torche plutôt qu'à la direction du joint comme il se devrait. Les détails de ce premier suiveur font partie du rapport du contrat CNRC OSD 82-00041.

Le suiveur de deuxième génération solutionnait les problèmes de balayage en proposant un prototype de détection infra-rouge à balayage électronique. Le système utilisait un faisceau de fibres optiques et une ligne de capteurs PBS indépendants. De plus, les degrés de liberté du système étaient augmentés à trois incluant deux translations orthogonales et une rotation. Les fibres optiques utilisées transmettaient la radiation IR jusqu'à une longueur d'onde de 2 microns. Il était nécessaire d'observer une région à haute température pour obtenir un signal adéquat. Par conséquent, la distance entre la ligne focalisée à l'avant du bain et le bain lui-même était très critique et de l'ordre du millimètre. Les mouvements inévitables du bain de fusion provoquaient de temps à autre de faux signaux, ce qui entrainait la perte du joint, en cours de suivi.

Pour régler ces problèmes il fût décidé d'étendre la bande passante jusqu'à 4.5 microns pour permettre la détection de température jusqu'à l'ambiante et augmenter la distance balayage-bain de fusion pour obtenir un suivi de joint plus fiable.

D'autres améliorations techniques ont aussi été apportées. En effet, le système de détection infra-rouge à éléments discrets a été remplacé par une caméra infra-rouge industrielle. Les dérives en température des détecteurs du suiveur de deuxième génération sont donc corrigées.

La caméra est interfacée à une carte MULTIBUS venant prendre place dans le contrôleur pour soudage adaptatif développé antérieurement par l'Institut de Soudage du Canada (1). Cette carte d'interface permet de numériser une image bidimensionnelle complète. On a donc à notre portée un système complet de vision infra-rouge complètement informatisé permettant l'acquisition et le traitement d'images. Finalement, ce système pourrait être utilisé pour des projets pilotes en industrie car il est basé sur une caméra portative industrielle et sur une armoire de contrôle robuste dédiée au soudage adaptatif.

(1)-- Contrat CNRC No. OSD84-00083

L'objet de la seconde phase du projet concerne la conception des logiciels d'applications nécessaires pour effectuer le suivi d'un joint bout à bout à bord droit à l'aide d'une caméra AGA THERMOVISION 110. Le matériel utilisé fut mis au point lors de la phase 1 de ce projet (1). Un système complet d'imagerie infra-rouge fut mis au point pour permettre l'étude des phénomènes thermiques en périphérie du bain de fusion. L'utilisation du faisceau de fibres optiques tel que décrit au prédécent rapport (1) a dû être abandonné en raison d'un bris mécanique.

Une étude extensive des images obtenues par focalisation directe du bain de fusion et de la zone environnante a été effectuée. Les résultats de cette étude ont confirmé l'hypothèse qu'une bande passante élargie permettrait de détecter le joint plus en avant du bain de fusion. Ils ont aussi mené à la conception et à la réalisation d'un logiciel de suivi de joints bout à bout à bords droits.

(1) Asservissement des paramètres en soudage à l'aide d'un système de vision passive (phase 1) contrat NO OSD 84-0 0300

## 2. INTEGRATION DU MATERIEL

### 2.1 Système optique:

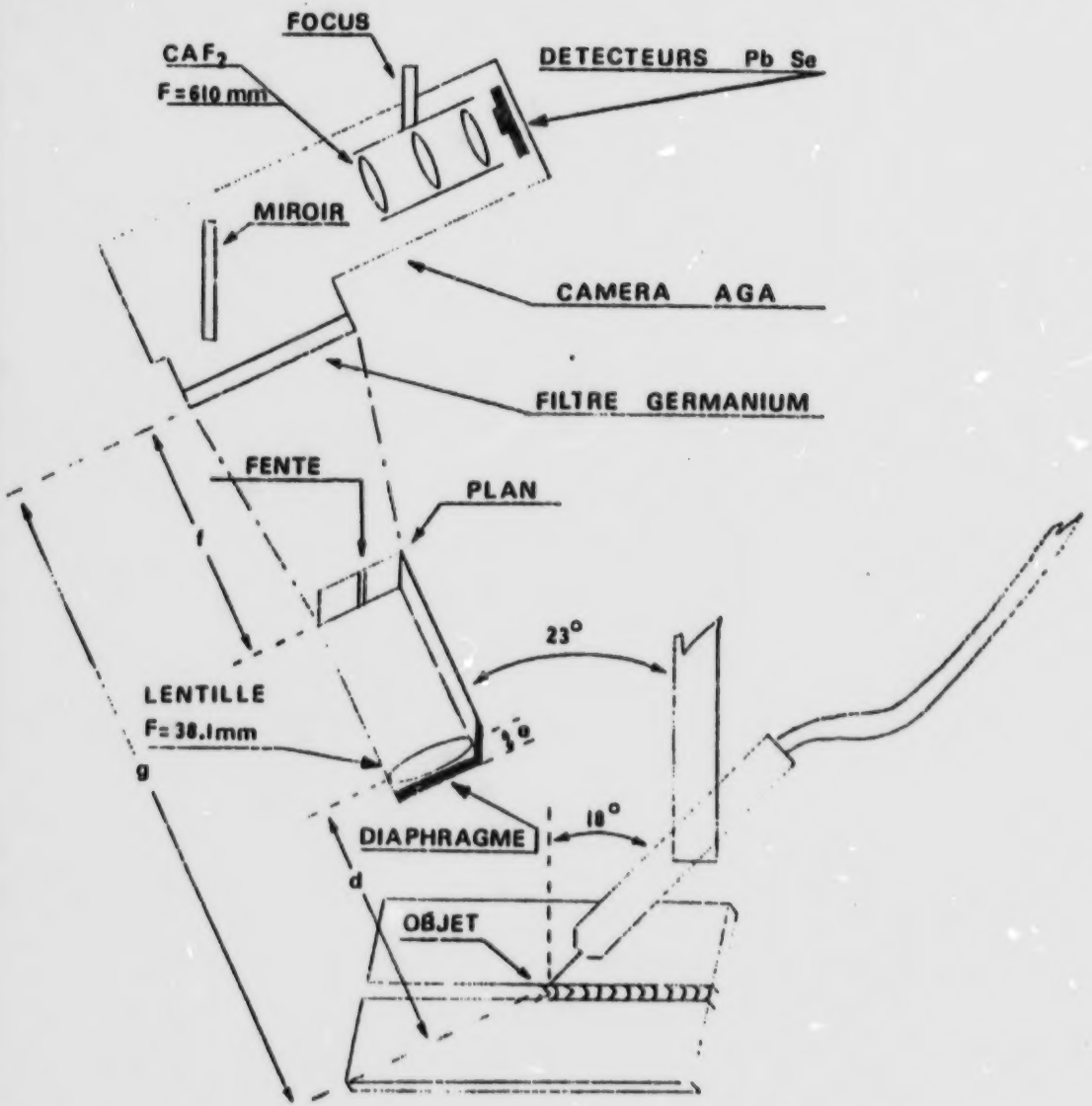
Certaines modifications au système optique ont du être apportées. L'information provenant du rayonnement des plaques à souder fut d'abord acheminée à la caméra AGA via un conduit de fibres optiques. La caractéristique principale de la fibre optique utilisée est de transmettre dans une région de 0,5 à 4 micro-mètres en présentant une atténuation de l'ordre du dixième de décibel par mètre. Il s'agit de fibres fabriquées à partir d'une forme de verre de fluorure de zirconium.

Ces fibres sont cependant excessivement fragiles. En effet, le canon à fibres qui fut construit ne supporta pas les manipulations que nécessitairent les premiers essais et fut détruit. Des modifications ont donc été apportées pour permettre une focalisation directe de la zone en avant du bain de fusion à l'aide de la caméra AGA et ce, sans faisceau de fibres.

La figure 2.1 montre un schéma représentatif de l'installation utilisée. Une caméra AGA thermovision modèle 110 est installée sur un support métallique solidaire de la torche à souder. Le système optique de la caméra AGA n'a pas été modifié de façon permanente dans le but d'utiliser cette même caméra pour d'autres applications ultérieurement. Une lentille au CaF2 de longueur focale égale à 610 mm a cependant été installée pour permettre une focalisation à une distance de 250 mm. Le système optique original de la caméra AGA ne permet pas de focaliser en deçà de 1 mètre. Un boitier optique incluant une lentille au CaF2 permet d'imager la scène dans un plan coïncidant avec le dessus du boitier. Le grossissement obtenu par l'utilisation du boitier optique est 1.4. Le champ de vision de l'ensemble optique est circulaire et d'un diamètre égal à 7.5 mm. Un diaphragme a été installé sur l'embout du

boitier optique du côté soudure pour atténuer le signal d'émission infra-rouge. Du côté caméra AGA trois épaisseurs de film kapton (120 microns) ont été disposés directement sur le filtre au germanium de façon à augmenter le contraste de l'image et produire un fond noir lorsqu'aucune irradiation infra-rouge n'est présente. La figure 2.2 montre le spectre de transmission infra-rouge moyen du kapton en fonction de la longueur d'onde de 1.4 à 5 microns. La figure 2.3 montre une photo de l'ensemble optique installé sur le banc de soudage GMAW.

Il est important de mentionner que les détecteurs de la caméra sont couplés capacitivement et que par conséquent les niveaux de couleurs des images produites ne représentent pas de véritables isothermes. On peut cependant s'affranchir de cette contrainte en disposant une fente dans le plan image intermédiaire du système optique situé au dessus du boîtier. On observe alors qu'une tranche de l'image seulement, mais les niveaux mesurés d'un essai à l'autre peuvent être comparés de façon absolue.




d=62mm

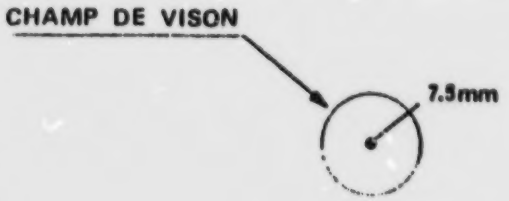
e=19mm

f=265mm

g=470mm

GROSSISSEMENT $\left(\frac{\text{PLAN}}{\text{OBJET}}\right)$ : 1.4

FENTE (LARGEUR) : 1mm




| | | | | | |
|---|---|---|---|---|---|
| | fig:2.1 Système Optique et Camera AGA | | | DR'N BY S. Landry | SCALE |
| | | | | CH'D BY P. Fafard Ing. | DWG. No. 2.1 |
| | MAT'L | No. REQ'D | CLASS | DATE | |

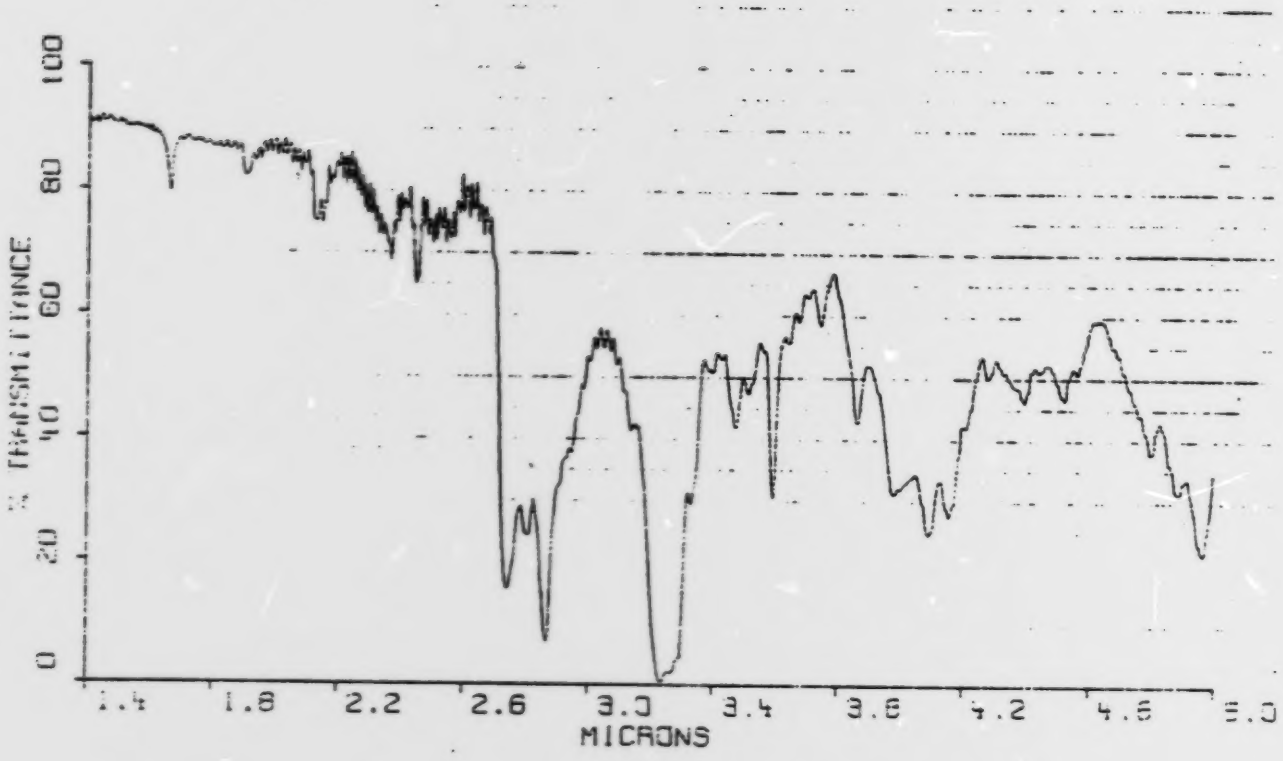



Figure 2.2: SPECTRE IR MOYEN (TRANSMISSION) DU KAPTON)

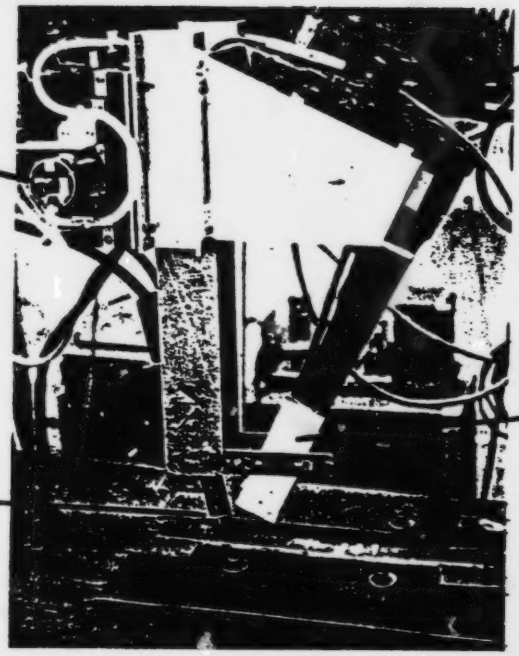



FIGURE 2.3: INSTALLATION DE LA CAMERA AGA THERMOVISION

# 3. LOGICIEL DU SYSTEME DE VISION INFRA-ROUGE

## 3.1 INTRODUCTION

Le but du projet étant d'effectuer un suivi de joint, nous avons dû au préalable nous munir d'outils afin d'analyser le procédé.

Dans une première phase, nous avons développé un logiciel d'imagerie PSEUDO.C. Ce programme affiche en temps réel, environ une image par seconde, les images acquises par le système de vision infra-rouge. La sortie est une image en pseudo-couleurs afin d'accentuer la perception des différentes températures sur les plaques durant le soudage.

Dans une deuxième phase, nous avons développé le programme de suivi de joint en tant que tel. Celui-ci est composé de deux modules SUIVI.C et SUITE.C.

La structure des deux programmes SUIVI.C et PSEUDO.C est essentiellement la même. Nous allons donc nous contenter d'expliquer celle du programme SUIVI.C.

## 3.2 STRUCTURE DU PROGRAMME *SUIVI.C.*

Le langage "C" possède une structure modulaire. Un programme est composé de modules ou fonctions qui font appel à d'autres fonctions. On peut donc facilement implanter une structure de type "TOP DOWN".

Le module principal se nomme main () et est commun à tout programme écrit en "C". Une liste des fonctions des modules SUIVI.C et SUITE.C est indiquée à la page suivante. Chaque fonction possède un niveau dans la structure du programme. Une fonction d'un certain niveau ne peut appeler que des fonctions d'un niveau plus bas.

ex: mode_colonne_zéro / niveau 2
appelle
écriture_mémoire / niveau 6

La description de chaque fonction est donnée en commentaire dans le fichier source du programme.

**********

LISTE DES FONCTIONS

**********

```
1----2----3----4----5----6----7----------------------> Niveau de la fonction

main()
                        acquisition_profil()
                        affiche_centre()
                        affiche_derivee()
                        affiche_profil()
                   analyse_profil()
              avance_compteur(compteur)
                        calcul_centre_hauteur()
                        calcul_commande_centre()
                        calcul_derivee()
                        decremente(compteur)
              demande_numero(nombre_digit)
         depart()
                                  ecriture_attribut_message(ligne, attribut)
                                  ecriture_message(ligne, numero_message)
                                  ecriture_mode(mode)
                             ecriture_numero(ligne, numero)
                                  ecriture_parametre(ligne, pmessage)
                        incremente(compteur)
              initialisation_cy512()
                        lancement_acquisition()
              lancement_commande()
                   memorise_profil(numero_profil)
         mode_colonne_profil()
         mode_colonne_zero()
         mode_direct()
         mode_ecriture_disque()
         mode_lecture_disque()
         mode_ligne_basse()
         mode_memorise()
         mode_offset()
         mode_rappel()
         mode_suivi()
         mode_taux_memorisation()
                   rappel_profil(numero_profil)
              reset_cy512()
              transfert_profil()
                                  vidange_message(ligne)
                                  vidange_parametre(ligne)
                        x_commande(commande)
                   x_deplacement(nombre_pas)
                        y_commande(commande)

1----2----3----4----5----6----7----------------------> Niveau de la fonction
```

## 3.3 ALGORITHME DE RECHERCHE DU JOINT.

L'algorithme de recherche du joint qui est utilisé par le programme SUIVI.C est essentiellement le même que dans nos travaux précédents.

Il s'agit de détecter la position des deux sommets du profil. Le centre du joint est alors égal au point à la mi-distance entre les deux sommets.

Pour ce faire, on procède à une dérivée filtrée du profil. Les transitions de cette dérivée nous indiquent alors les deux sommets.

## 4. ESSAIS ET INTERPRETATION DES RESULTATS.

Le système d'acquisition d'images consiste en une caméra infra-rouge AGA modèle 110 interfacée à un ordinateur tel que décrit dans le rapport de l'Institut de Soudage du Canada RC-263 intitulé <<Asservissement des paramètres en soudage à l'aide d'un système de vision passive (phase 1)>>.

La première étape a consisté à étudier la zone en périphérie du bain de fusion, plus précisément la zone en avant du bain de fusion. Il s'agissait d'abord de vérifier l'hypothèse selon laquelle les isothermes à la surface des plaques à souder en soudage GMAW pouvaient être détectées et analysées à une distance de quelques mm en avant du bain de fusion. L'utilisation du système développé autour de la caméra AGA permet de capter l'émission infra-rouge jusqu'à une longueur d'onde de 5 micromètres. Le système précédent (1) était limité à une détection à 2 micromètres et exigeait un ajustement précis de l'optique de façon à focaliser sur une ligne passant par les cornes du bain de fusion. Plusieurs images ont donc été numérisées en cours de soudage de façon à bien comprendre les phénomènes présents autour du bain de fusion.

Les tableaux 1 et 2 donnent le détail des paramètres utilisés pour les différents essais. La figure 4.0 donne la représentation des cotes utilisées aux tableaux 1 et 2. Les essais ont été regroupés en cinq sections:

1- Essais sur un joint bout à bout avec préparation en V.(pleine image)

2- Essais sur un joint bout à bout à bords droits (pleine image)

1: Contrat USD. 31950-3-0008

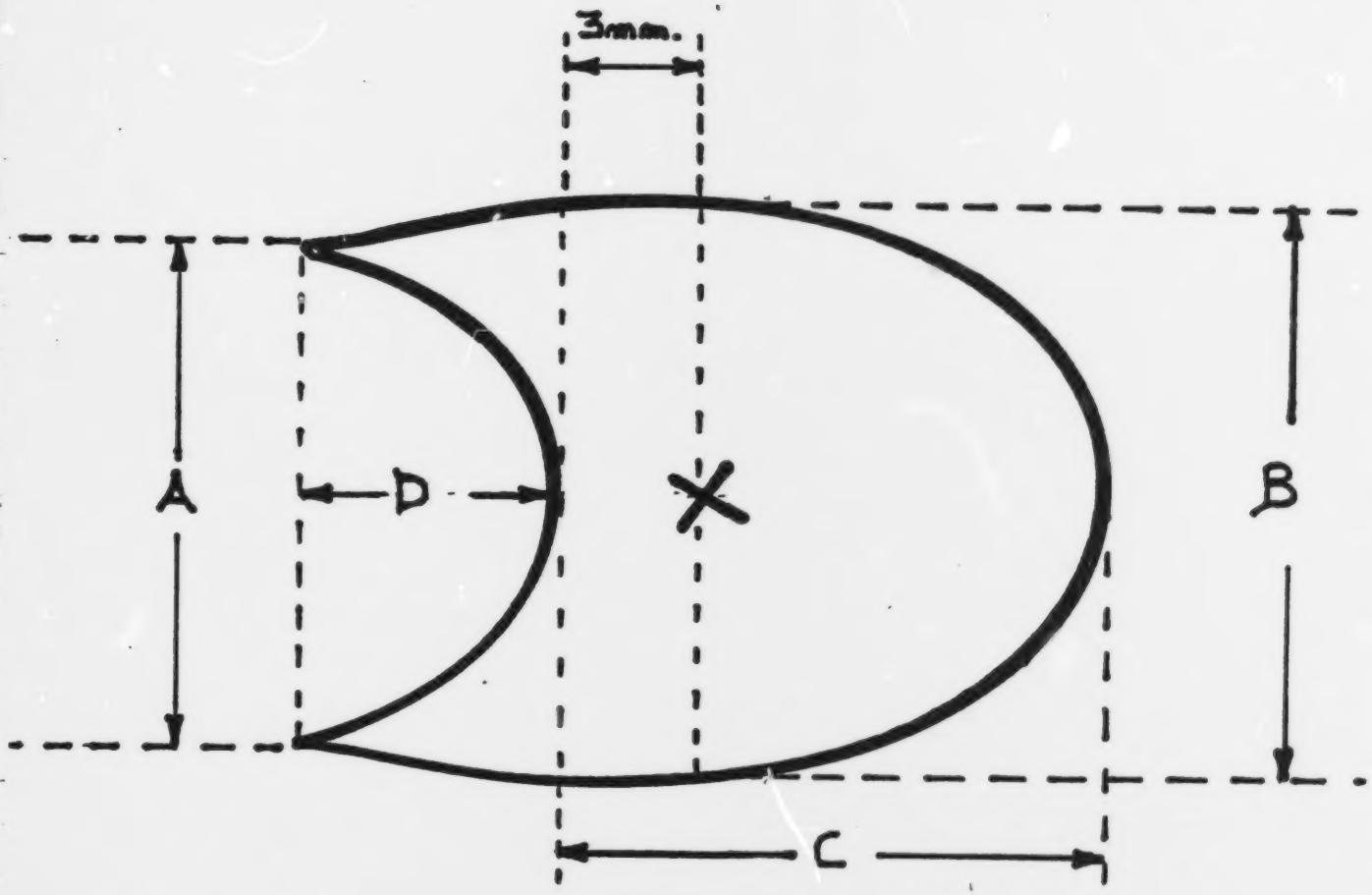



FIGURE 4.0: DIMENSIONS D'UN BAIN DE FUSION

| ESSAIS # | VITESSE SOUDAGE mm/sec Vs | VITESSE DEVIDAGE mm/sec Vd | TENSION DE SOUDAGE volts U | ECARTEMENT MESURE mm A | BAIN LONGUEUR mm C | BAIN LARGEUR mm B | DIMENSION DES CORNES mm D |
|---|---|---|---|---|---|---|---|
| 000 | 2.9 | 84.0 | 16.3 | 0.4 | 10.7 | 7.0 | 0.0 |
| 001 | 2.9 | 84.0 | 16.3 | 0.8 | 11.0 | 7.0 | 0.0 |
| 002 | 2.9 | 84.0 | 16.3 | 1.0 | 11.0 | 7.2 | 0.0 |
| 003 | 2.9 | 84.0 | 16.3 | 1.4 | 10.4 | 6.7 | 0.5 |
| 004 | 2.9 | 84.0 | 16.3 | 2.2 | 10.5 | 6.4 | 0.5 |
| 005 | 2.9 | 84.0 | 16.3 | 2.8 | 6.4 | 6.4 | 2.9 |
| 008 | 2.9 | 84.0 | 16.3 | 1.5 | 10.4 | 7.3 | 0.3 |
| 011 | 2.9 | 84.0 | 16.3 | 0.9 | 10.7 | 6.2 | 0.0 |
| 012 | 2.9 | 84.0 | 16.3 | 1.5 | 11.0 | 9.2 | 0.5 |
| 014 | 2.9 | 84.0 | 16.3 | 2.1 | 10.7 | 8.8 | 0.5 |
| 9 | 2.9 | 84.0 | 16.3 | 2.0 | 10.6 | 8.8 | 0.5 |
| 11a | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11b | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11c | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11d | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11e | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11f | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11g | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 11h | 2.9 | 84.0 | 16.3 | 2.1 | 9.2 | 6.9 | 0.8 |
| 14 | 4.9 | 97.4 | 17.3 | 1.0 | 8.0 | 6.0 | 0.5 |
| 14a | 1.0 | 67.6 | 15.7 | 2.5 | 11.7 | 10.2 | 0.0 |
| 14b | 1.0 | 67.6 | 15.7 | 2.5 | 10.1 | 7.8 | 1.1 |
| 14c | 2.9 | 84.0 | 16.3 | 2.5 | 10.1 | 8.0 | 1.1 |
| 14d | 4.9 | 97.4 | 17.3 | 1.0 | 8.0 | 6.0 | 0.5 |

Tableau 1.

Essais sur joints bout à bout avec préparation en V

| ESSAIS # | VITESSE SOUDAGE mm/sec Vs | VITESSE DEVIDAGE mm/sec Vd | TENSION DE SOUDAGE volts U | ECARTEMENT MESURE mm A | BAIN LONGUEUR mm C | BAIN LARGEUR mm B | DIMENSION DES CORNES mm D |
|---|---|---|---|---|---|---|---|
| 017 | 2.0 | 89.0 | 16.8 | 2.5 | 11.0 | 7.6 | 1.0 |
| 018 | 2.0 | 89.0 | 16.8 | 2.5 | 11.0 | 7.6 | 1.0 |
| 019 | 2.0 | 89.0 | 16.8 | 2.5 | 11.0 | 7.7 | 1.0 |
| 020 | 2.0 | 89.0 | 16.8 | 2.5 | 11.0 | 7.8 | 1.0 |
| 021 | 2.0 | 89.0 | 16.8 | 2.5 | 10.6 | 8.0 | 1.0 |
| 022 | 2.0 | 89.0 | 16.8 | 2.5 | 11.0 | 8.2 | 1.0 |
| 023 | 2.0 | 89.0 | 16.8 | 2.5 | 11.4 | 8.4 | 1.0 |
| 024 | 2.0 | 89.0 | 16.8 | 2.5 | 12.0 | 8.4 | 1.0 |
| 047 | 4.2 | 126.0 | 17.4 | 0.7 | 13.7 | 7.1 | 0.5 |
| 049 | 2.2 | 105.0 | 17.4 | 0.4 | 16.0 | 10.0 | 0.7 |
| 051 | 1.5 | 89.0 | 16.8 | 2.7 | 11.2 | 8.6 | 0.9 |
| 053 | 2.5 | 89.0 | 16.8 | 2.7 | 9.5 | 8.0 | 1.0 |
| 054 | 3.8 | 89.0 | 16.8 | 2.7 | 4.4 | 5.0 | 2.9 |
| 043 | 2.5 | 117.0 | 18.4 | ANGLE | 11.1 | 6.8 | 0.0 |
| 045 | 2.5 | 117.0 | 18.4 | LAP | 12.4 | 8.0 | 0.0 |

Tableau 2.

Essais sur joints bout à bout à bords droits

3- Essais sur un joint bout à bout avec préparation en V (utilisation d'une fente)

4- Essais sur un joint bout à bout à bords droits (utilisation d'une fente)

5- Essais sur un joint à recouvrement et sur un joint d'angle (pleine image)

## 4.1 Essais sur un joint bout à bout avec préparation en V (pleine image)

Cette section regroupe les essais (000, 001, 002, 003, 004, 005, 008, 011, 012, 014, 9, 14 et 14a à 14d) En plus des variables indiquées au tableau 1 , les informations suivantes complètent la description des essais.

- Inclinaison de la torche de 18 degrés vers l'avant
- Direction du soudage (en poussant)
- Position à plat
- Fil NS-116 0.9mm non cuivré
- Gaz utilisé: mélange 90% Ar et 10% CO2
- Débit du gaz: 0.7 mètre cube/heure
- Coupons: Acier au carbone A-36
- Epaisseur des tôles: 5mm

Chaque image fut prise en cours de soudage et mémorisée sur disquette. Par la suite le logiciel d'acquisition d'image tel que décrit à la section 3 est utilisé pour afficher à l'écran une image utilisant 16 pseudo-couleurs. La figure 4.1 montre l'image originale telle que photographiée sur un écran couleur IBM. Il faut noter sur cette image un champ de vision ovale dû à l'affichage sur écran cathodique. Le rapport d'aspect du champ circulaire de 7.5 mm de diamètre n'est pas conservé. L'échelle en abscisse est double de celle en ordonnée. Ce phénomène se retrouve sur toutes les images incluses dans ce rapport. Les sections foncées à droite et à gauche de l'image sont dues au principe de fonctionnement même de la caméra AGA thermovision modèle 110.

Les détecteurs au PbSe sont couplés capacitivement. Pour une ligne choisie une zone chaude sera donc toujours précédée et suivie d'une zone froide. Pour plus de détails le lecteur est référé au rapport RC-263 de l'Institut de Soudage du Canada concernant le contrat OSD84-00300 intitulé <<Asservissement des

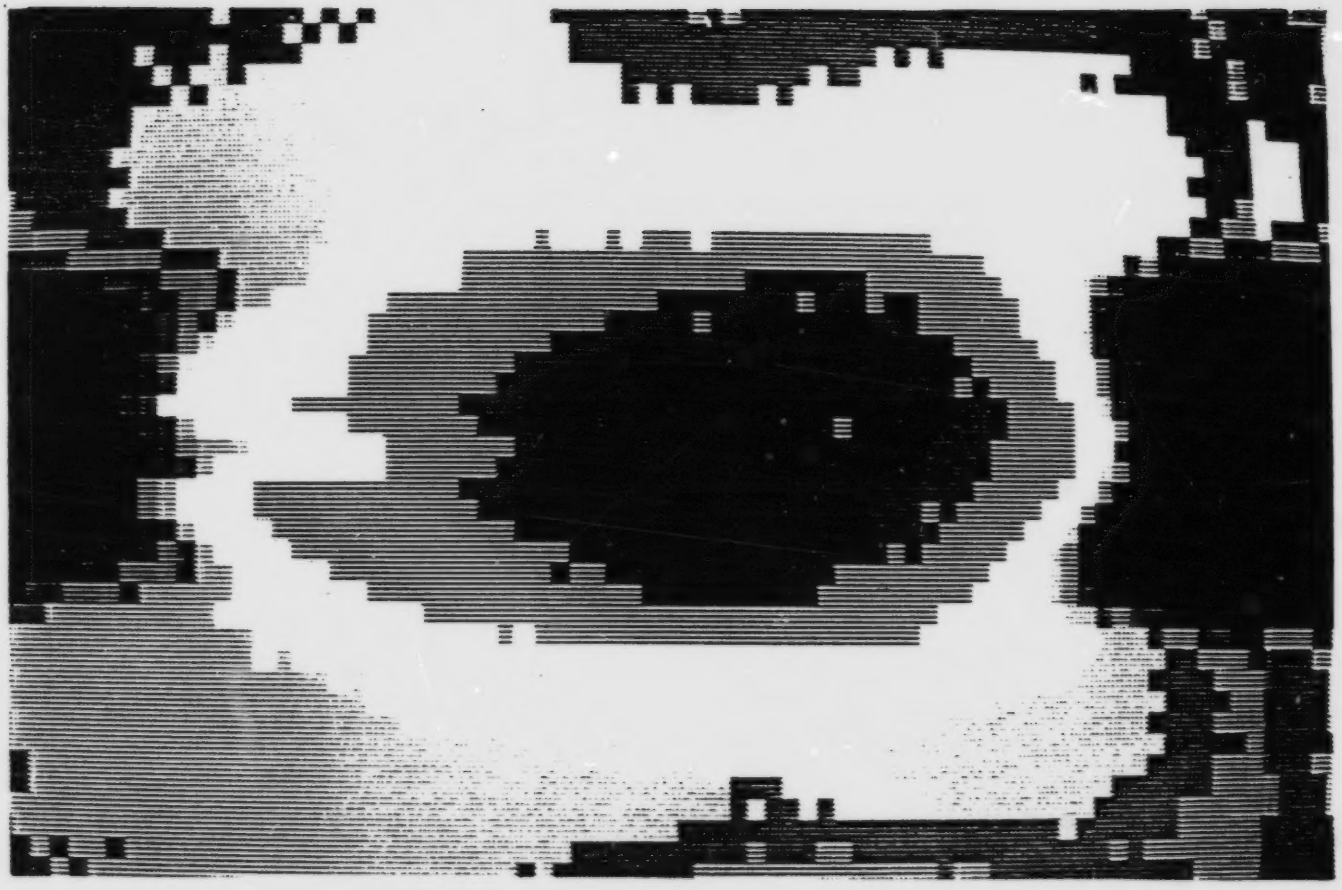


FIGURE 4.1: ECHANTILLON D'UNE IMAGE

paramètres en soudage à l'aide d'un système de vision passive (phase 1)>>.

Pour chacuns des essais répertoriés aux tableaux 1 et 2, un montage photographique a été effectué dans le but d'enlever l'information non utile de chacune des images. Une échelle graduée en millimètres est disponible et l'emplacement ainsi que les dimensions du bain de fusion et de l'écartement du joint ont été rajoutés pour obtenir une meilleure vue d'ensemble.

L'essai 000 présente des tôles avec écartement de 0.4 mm. La figure 4.2 montre une image typique obtenue en cours de soudage. Le signal obtenu ne permet pas de localiser clairement le joint à souder.

L'essai 001 présente un écartement des tôles de 0.8 mm. Dans ce cas le joint est visible jusqu'à 4 mm en avant du bain de fusion. Le système d'affichage graphique avec pseudo-couleurs permet de visualiser trois différentes isothermes. La figure 4.3 montre l'image obtenue. L'essai 002 (figure 4.4 ) montre une préparation en V avec écartement de 1 mm. Le joint est bien visible et le signal est utilisable jusqu'à 4.5 mm. en avant du bain. Le signal optimum se retrouve à environ 3 mm. en avant du bain de fusion. L'essai 003 (figure 4.5 ) montre un écartement de 1.4 mm. La région de mesure optimale se situe à 3 mm. devant le bain. Les essais 004 (figure 4.6 ) et 005 (figure 4.7 ) présentent des écartements de 2.2 et 2.8 mm respectivement. Dans chacun de ces cas le joint est facilement identifiable. Les essais 008 (figure 4.8 ) 011 (figure 4.9 ) 012 (figure 4.10 ) et 014 (figure 4.11 ) montrent des écartements variant de 0.9 à 2.1 mm. Dans chacun de ces cas un signal clair permet de localiser le joint à souder.

Les essais 14b (figure 4.23 ), 14c (figure 4.24 ) et 14 (figure 4.21 ) montrent l'effet de la vitesse de soudage (VS) sur les isothermes détectées. Pour l'essai 14b (figure 4.23 ) la vitesse est lente (1 mm./s), le joint est détectable jusqu'à 7 mm. en avant du bain de fusion. Pour l'essai 14c (figure 4.24 ) la vitesse est augmentée à 2.9 mm./s, un resserrement des isothermes peut déja être remarqué, le joint est localisable jusqu'à environ 5 mm. en avant du bain de fusion.

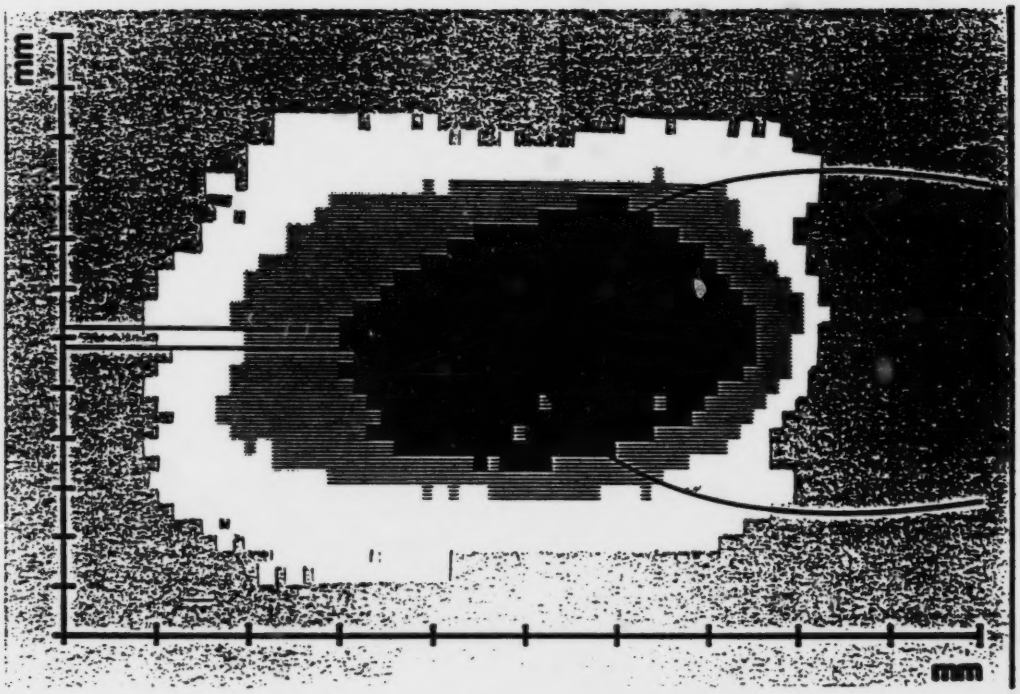



FIGURE 4.2: ESSAI 000

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

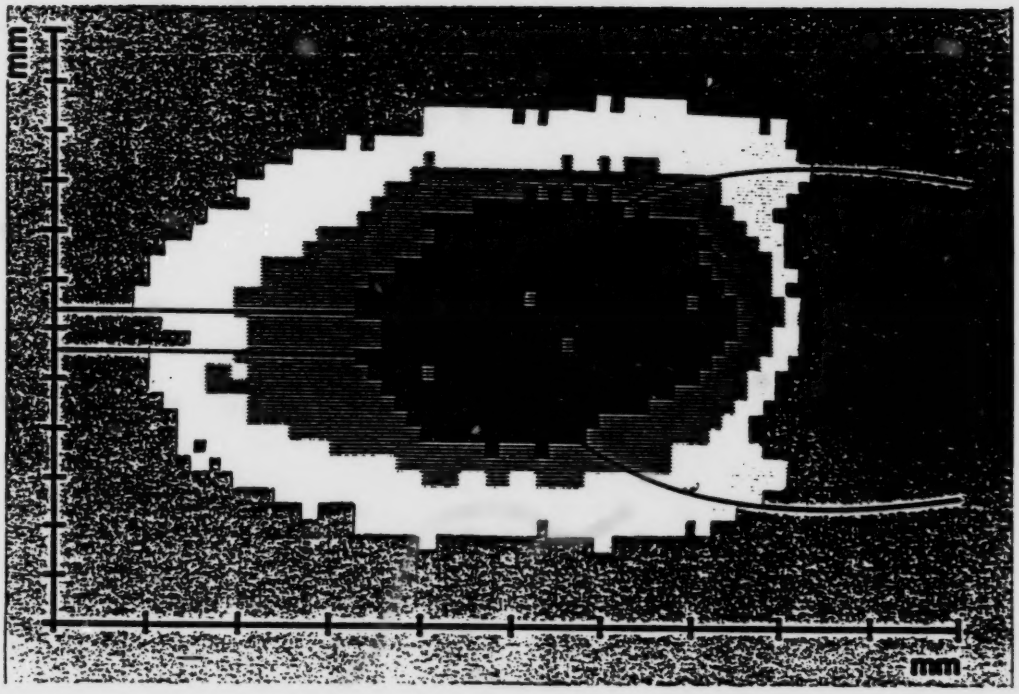



FIGURE 4.3: ESSAI 001

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

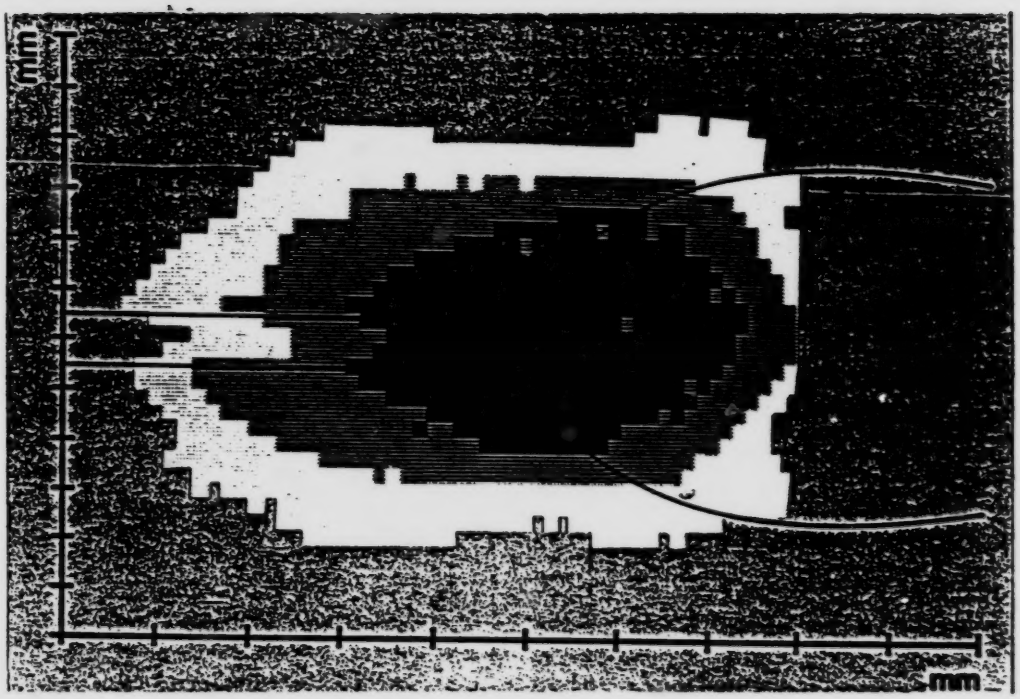



FIGURE 4.4: ESSAI 002

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

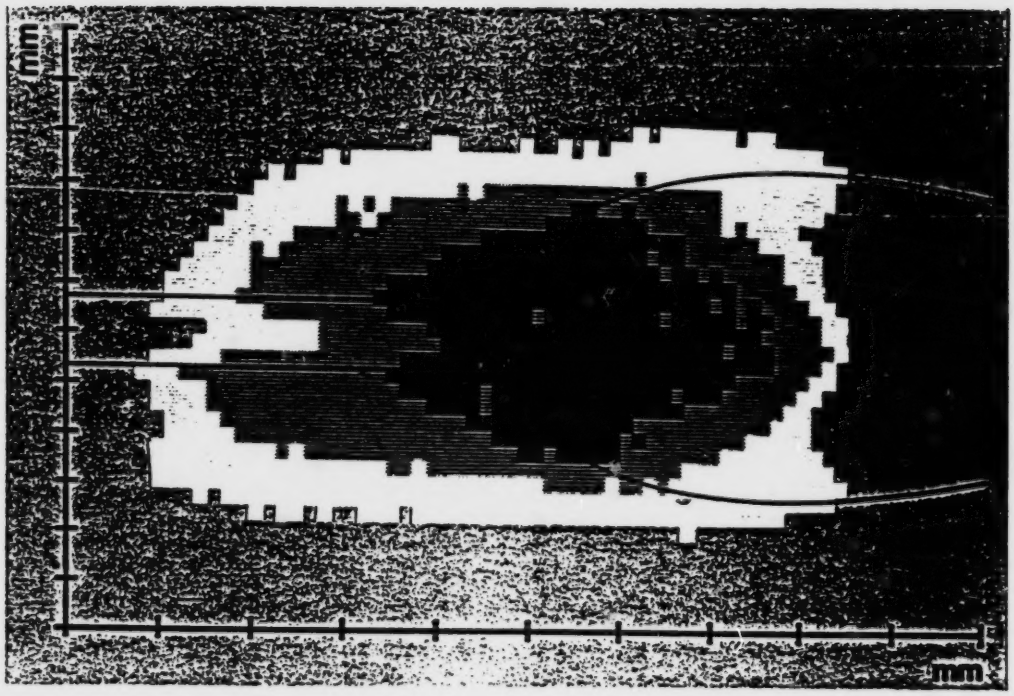



FIGURE 4.5: ESSAI 003

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

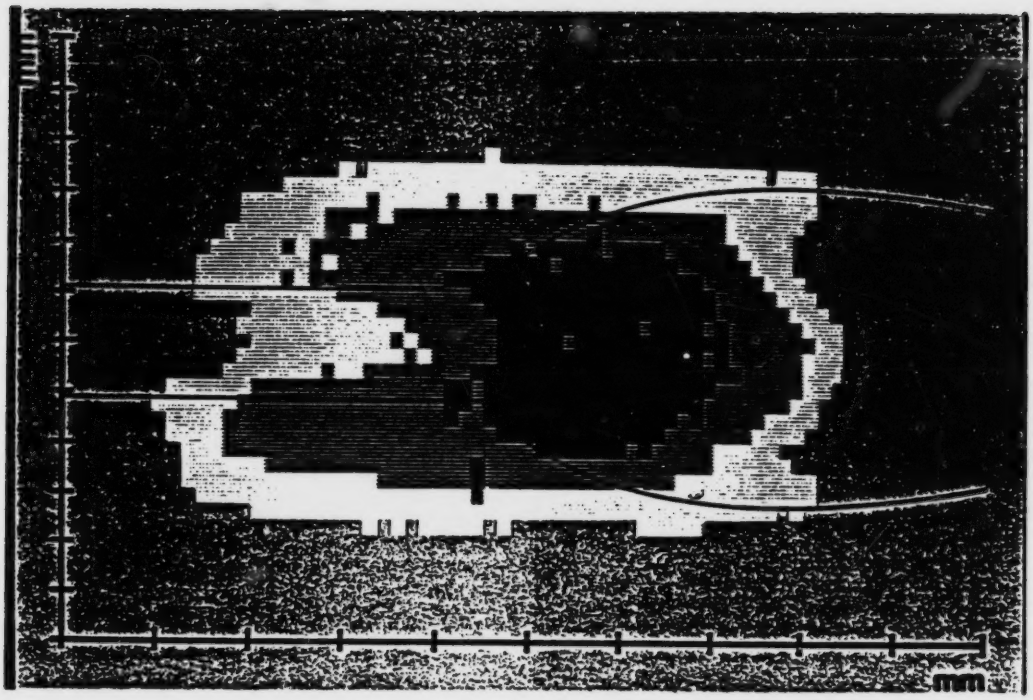



FIGURE 4.6: ESSAI 004
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

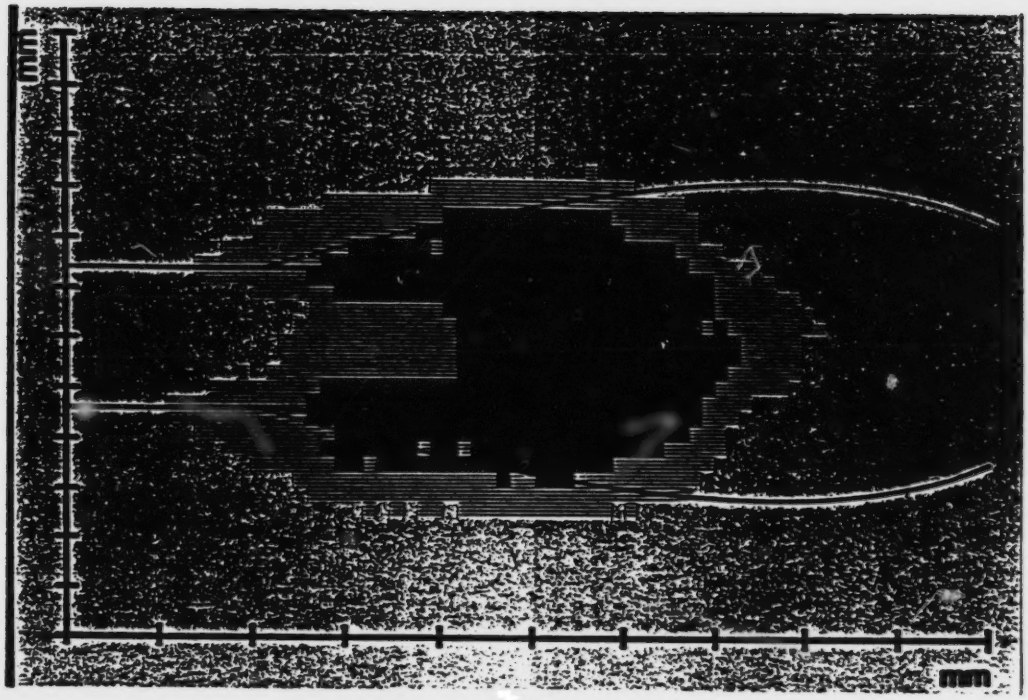



FIGURE 4.7: ESSAI 005

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

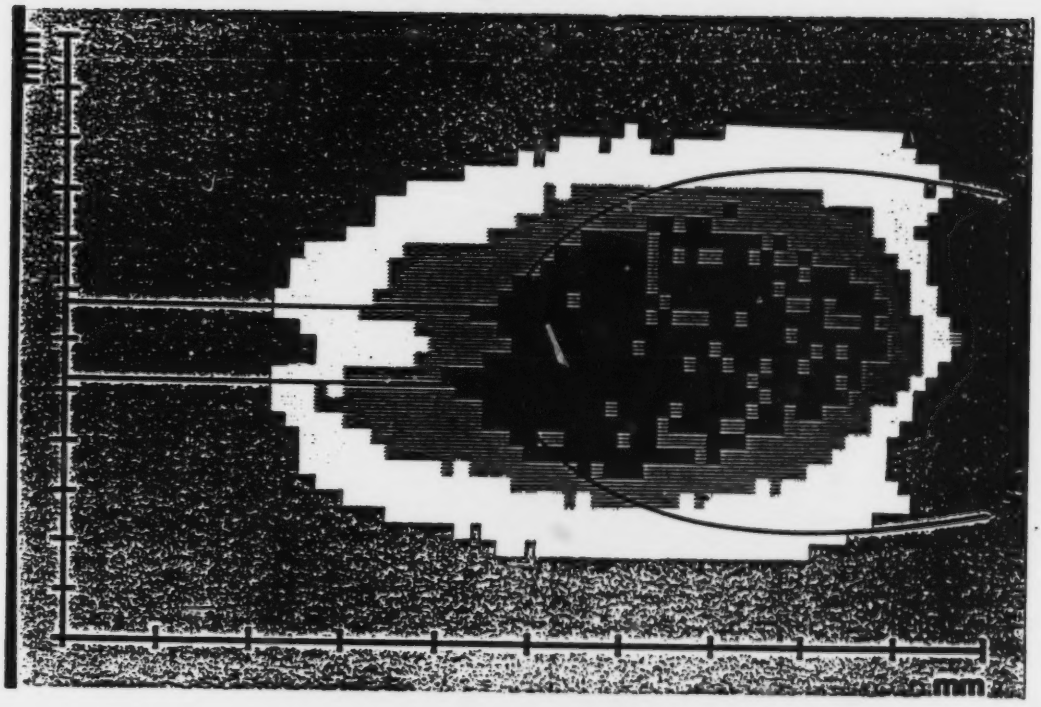



FIGURE 4.8: ESSAI 008
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

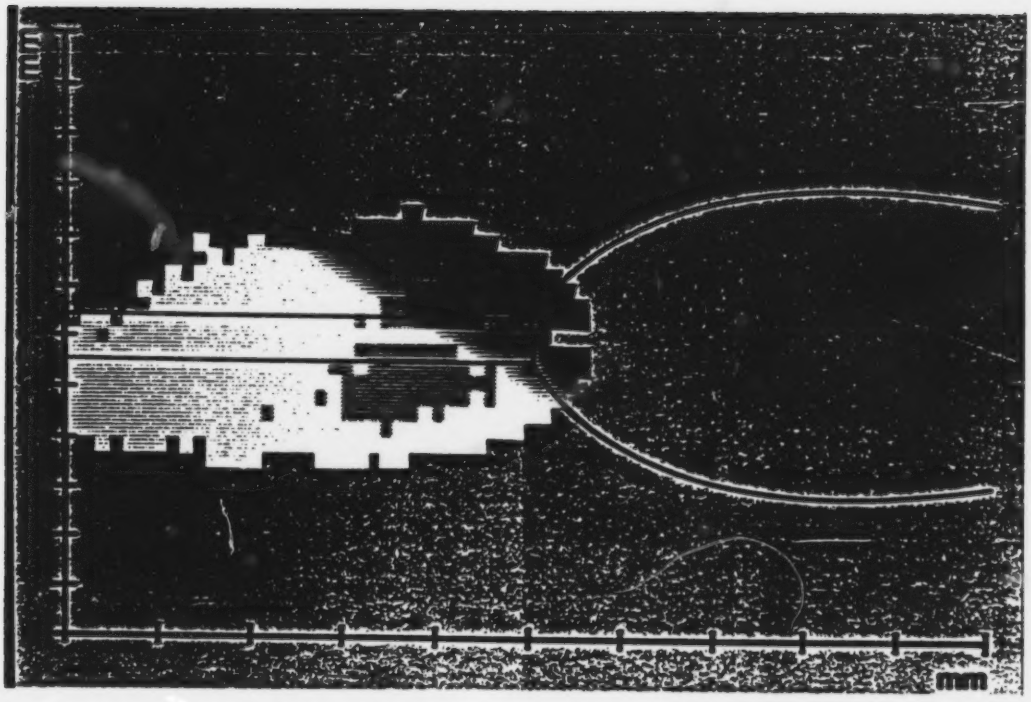



FIGURE 4.9: ESSAI 011
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

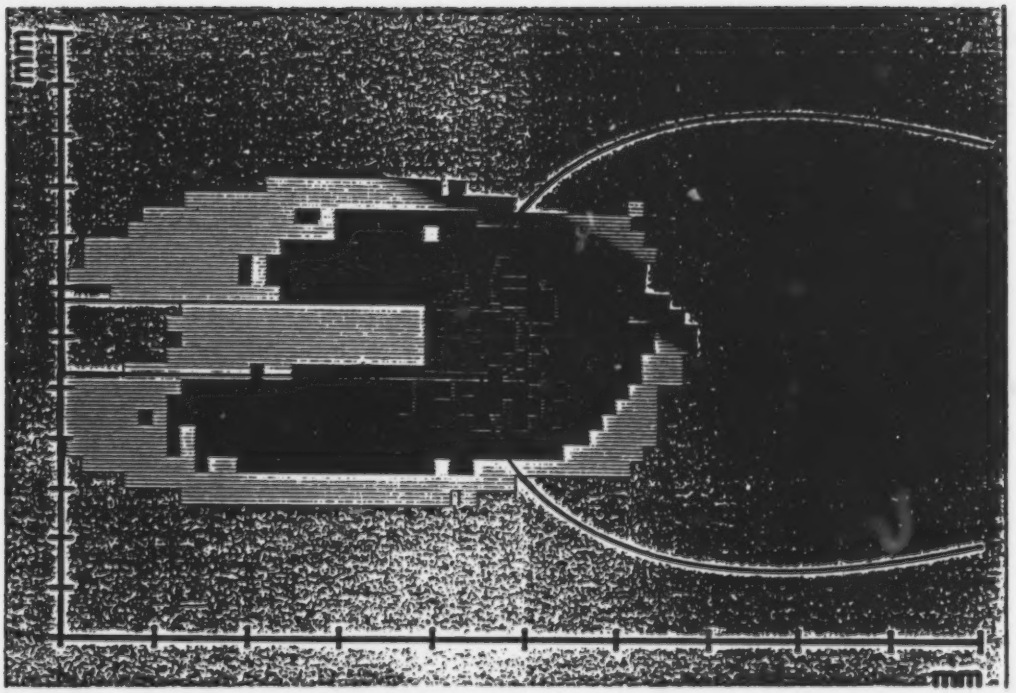


FIGURE 4.10: ESSAI 012

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

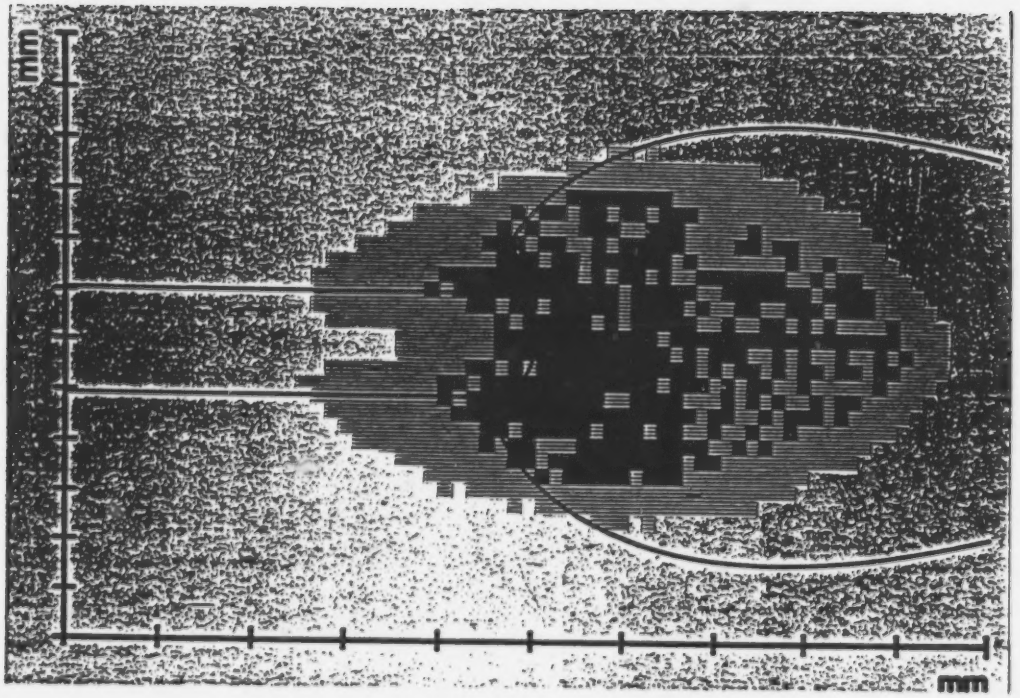



FIGURE 4.11: ESSAI 014

(JOINT BOUT A BOUT AVEC PREPARATION EN V)

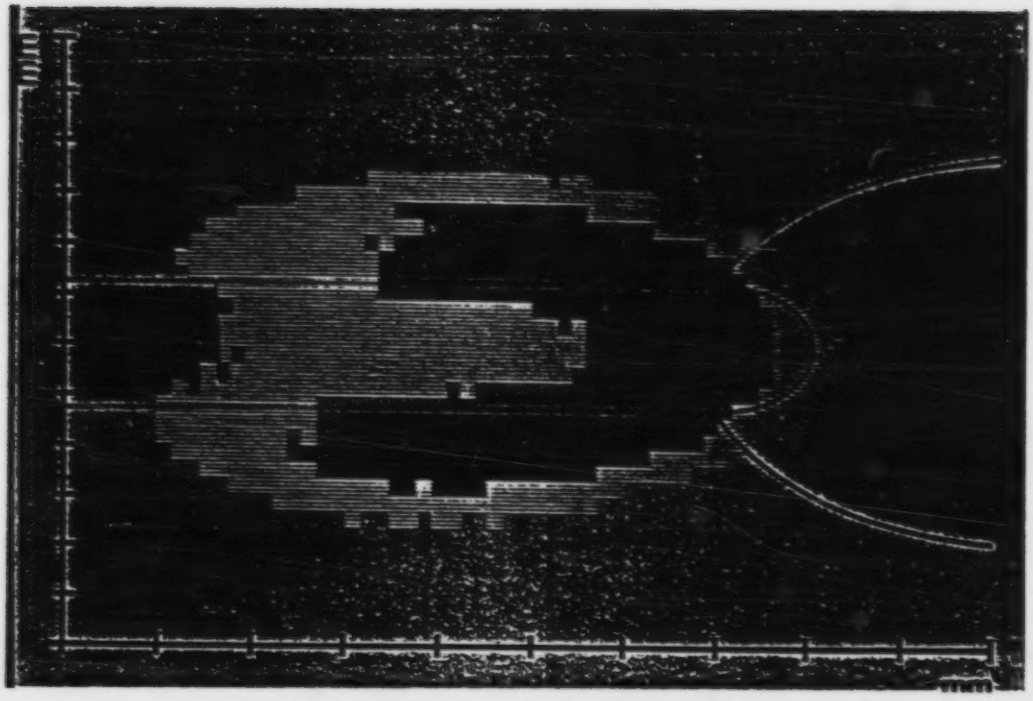


FIGURE 4.23: ESSAI 14B
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

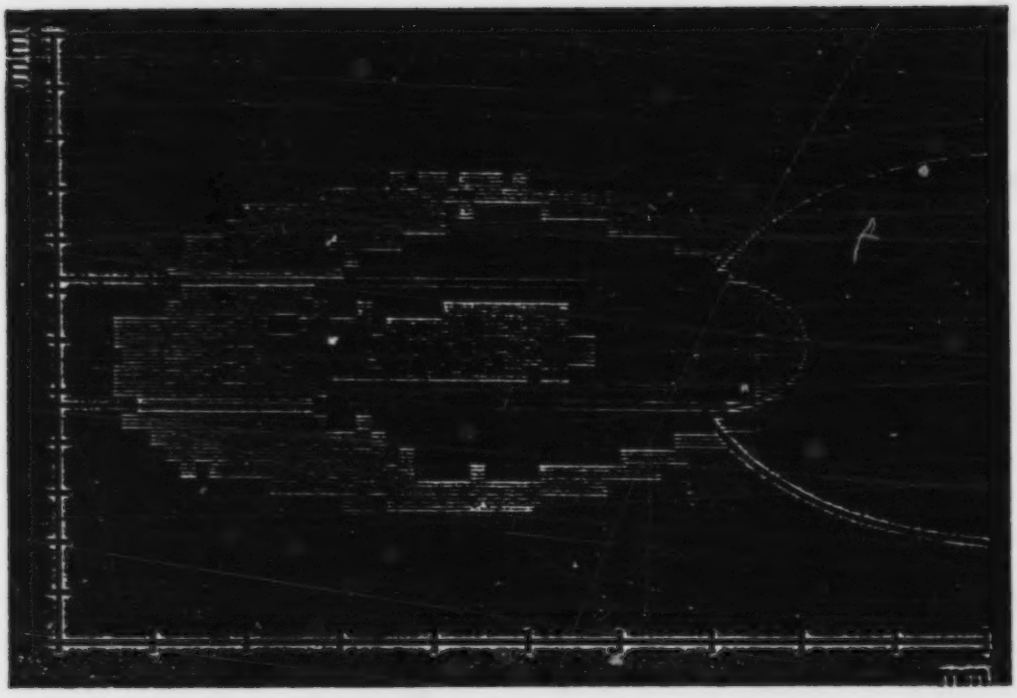


FIGURE 4.24: ESSAI 14C
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

Finalement l'essai 14 (figure 4.21 ) présente une vitesse de 4.9 mm. Les isothermes sont de plus en plus resserrées sur le bain. Le joint peut être détecté à peine 3 mm. en avant du bain. Dans ce cas-ci, l'écartement des plaques a du être ramené à 1 mm. pour pouvoir obtenir une soudure saine. Il n'était pas possible de souder le joint avec écartement de 2.5 mm.

Dans le même ordre d'idée le groupe d'essais 14a (figure 4.22 ), 9 (figure 4.12 ) et 14d (figure 4.25 ) montre une visualisation directe du bain de fusion et des isothermes à proximité pour trois vitesses de soudage. Le même effet de resserrement des isothermes est observé lorsque la vitesse de soudage croît.

Il est important de préciser que d'un essai à l'autre, les paramètres de soudage furent ajustés pour optimiser la qualité de la soudure. Cependant l'énergie linéaire de soudage n'est pas nécessairement constante.

Pour conclure cette section, les joints bout à bout avec préparation en V peuvent être détectés pour des écartements variant de 0.8 mm. à 2.8 mm. La région optimale de détection se situe à 3 mm. en avant du bain.

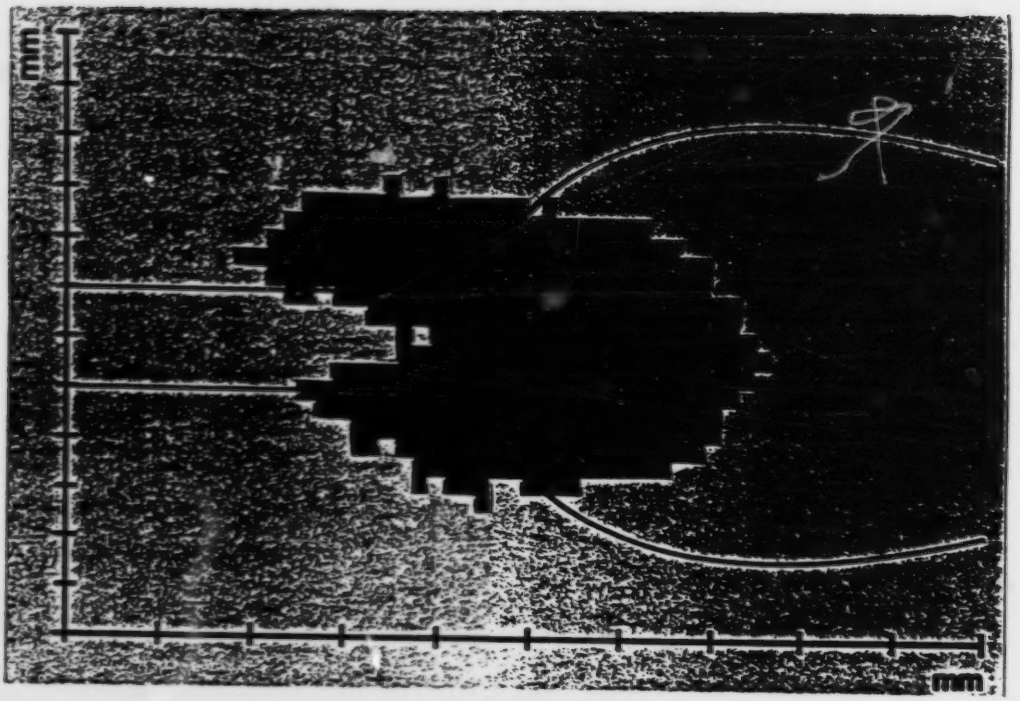



FIGURE 4.12: ESSAI 9
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

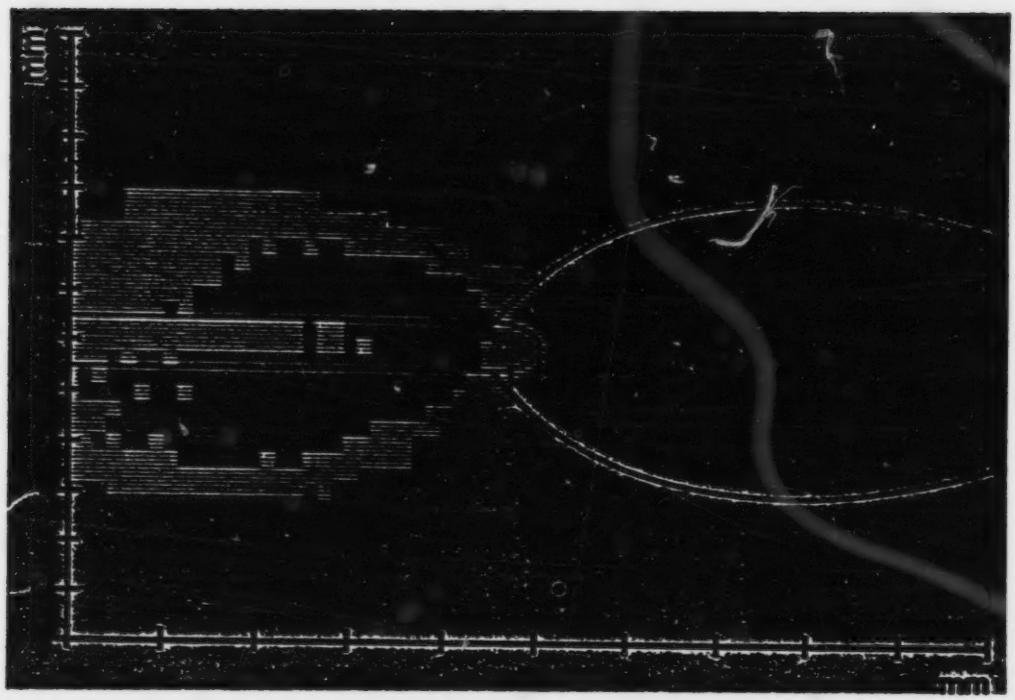


FIGURE 4.21: ESSAI 14
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

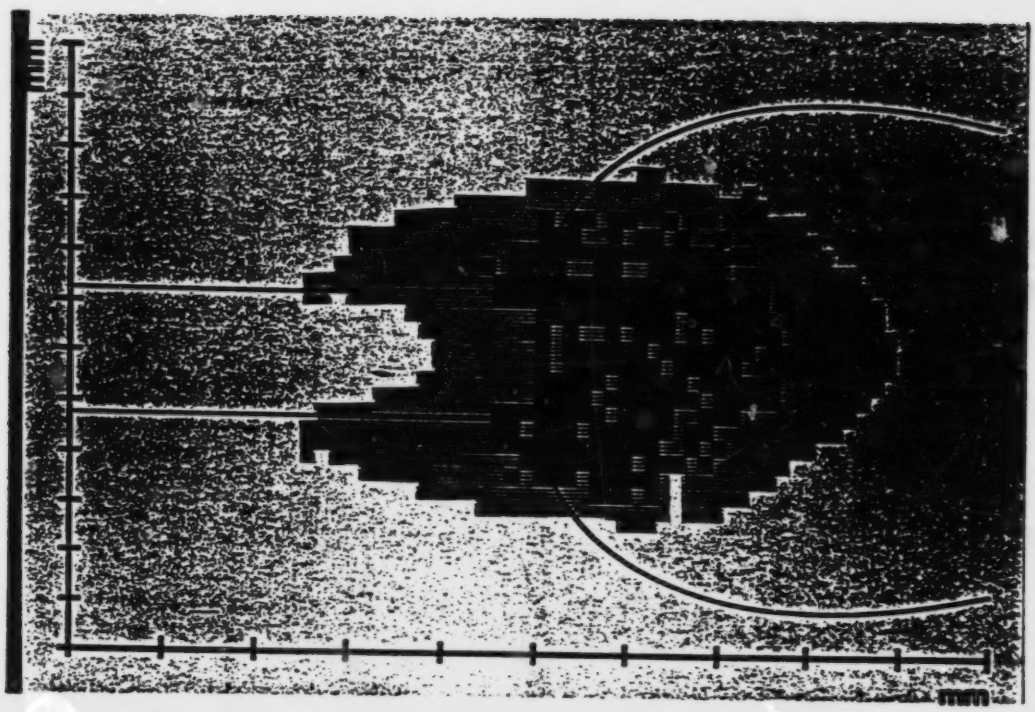



FIGURE 4.22: ESSAI 14A
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

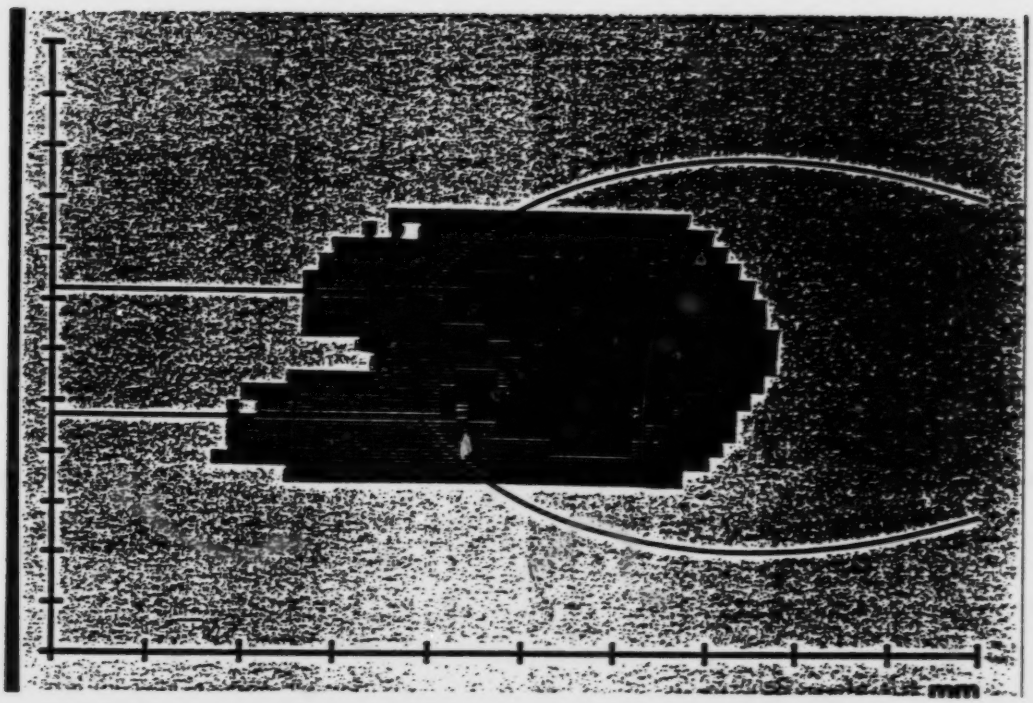


FIGURE 4.25: ESSAI 14D
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

## 4.2 Essais sur un joint bout à bout à bords droits (pleine image)

Cette section regroupe les essais 047, 049, 051, 052, 054. L'essai 047 (figure 4.34 ) présente un écartement de 0.4 mm. Les cornes du bain de fusion ont environ 0.5 mm. Le joint n'est pas visible. L'essai 049 (figure 4.35 ) montre un écartement de 0.8 mm. La région allant de 1.2 à 5 mm. à l'avant du bain de fusion permet de localiser le joint. Les essais 051 (figure 4.36 ) et 053 (figure 4.37 ) montrent les isothermes obtenues pour un écartement de 2.7 mm. Avec vitesse lente de 1.5 mm./s le signal permet de localiser le joint dans une région allant de 2.3 à 4.5 mm. en avant du bain de fusion. Avec une vitesse plus rapide de 2.5 mm./s (figure 4.37 ) les isothermes détectées sont davantage resserrées sur le bain et un signal peut être détecté dans une région allant de 1.8 à 3 mm. seulement en avant du bain de fusion. En effet l'isotherme mitoyenne s'est rapproché de 0.5 mm. du bain de fusion lorsqu'on compare les essais 051 et 053. L'essai 054 (figure 4.38 ) montre davantage l'effet de resserrement des isothermes pour une vitesse de 3.8 mm./s. L'isotherme mitoyenne s'est rapprochée du bain d'une distance de 1.3 mm. par rapport à l'essai 053 et à 2 mm. par rapport à l'essai 051. En conclusion de cette section, des joints bout à bout à bords droits avec écartement inférieur à 1 mm. peuvent être détectés, cependant pour une bonne fiabilité des écartements minimums de 1 mm. sont requis.

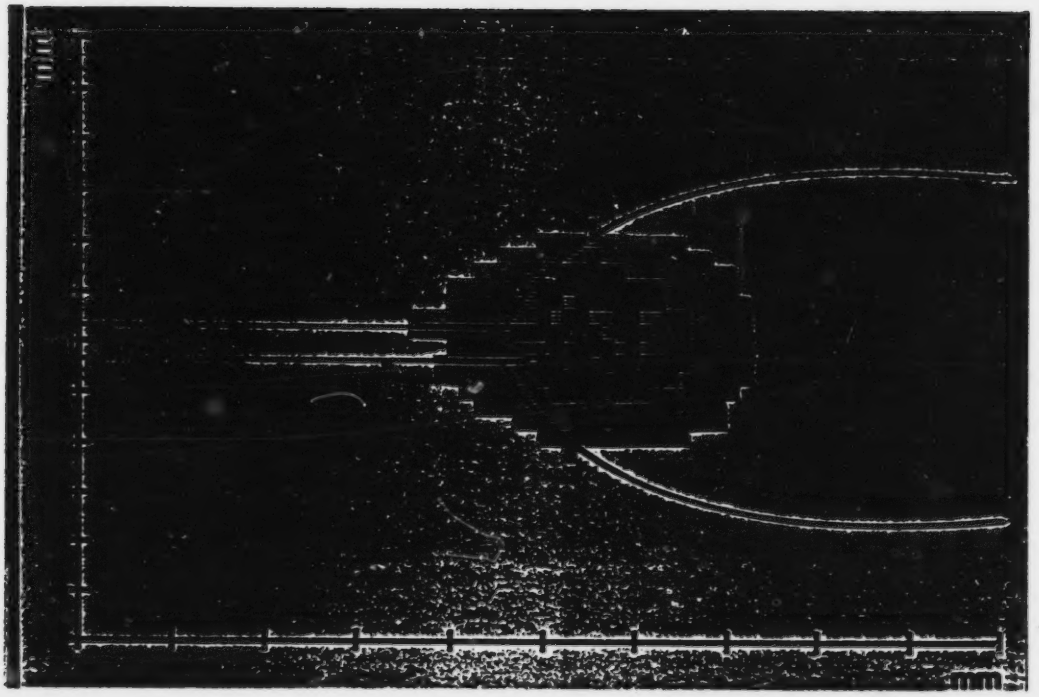


FIGURE 4.34: ESSAI 047
(JOINT BOUT A BOUT A BORDS DROITS)

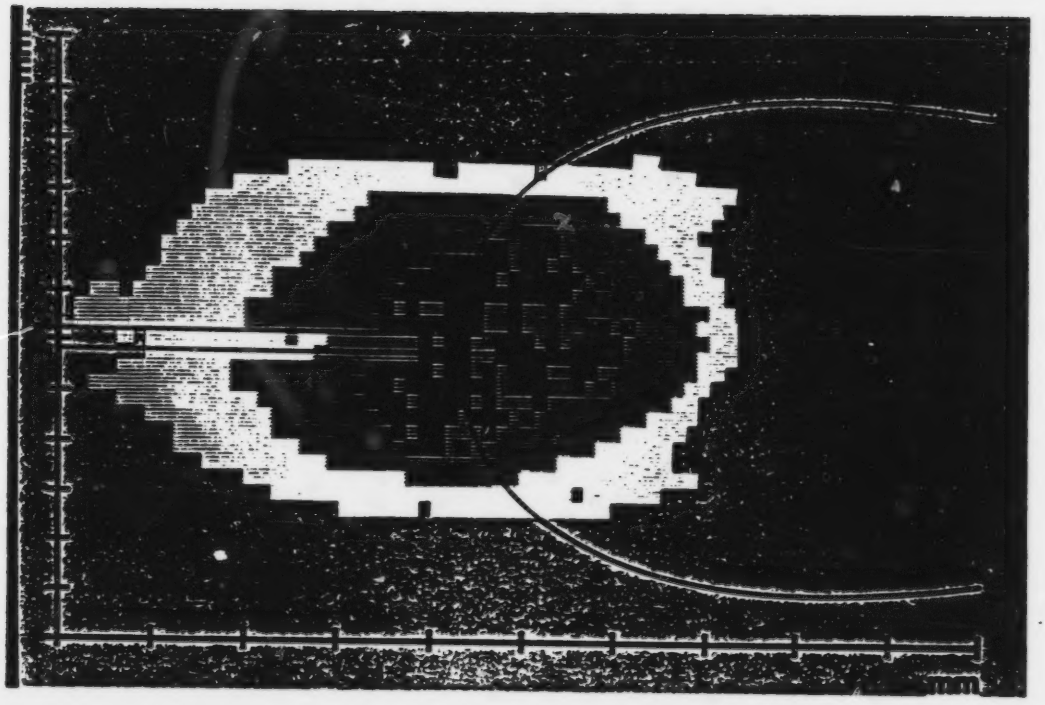


FIGURE 4.35: ESSAI 049
(JOINT BOUT A BOUT A BORDS DROITS)

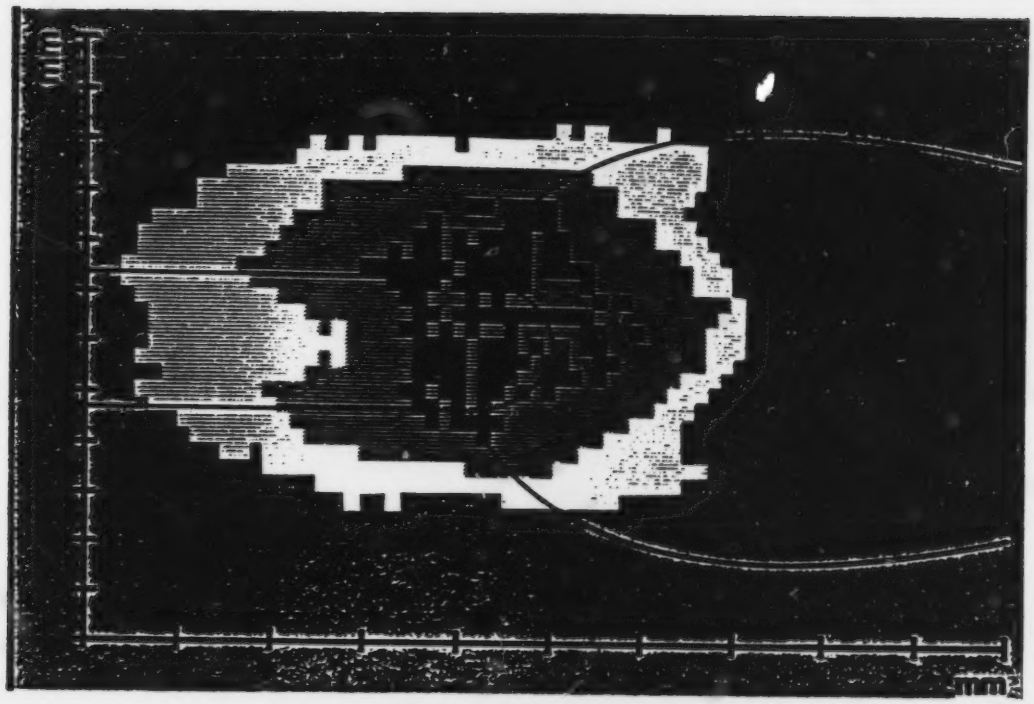


FIGURE 4.36: ESSAI 051
(JOINT BOUT A BOUT A BORDS DROITS)

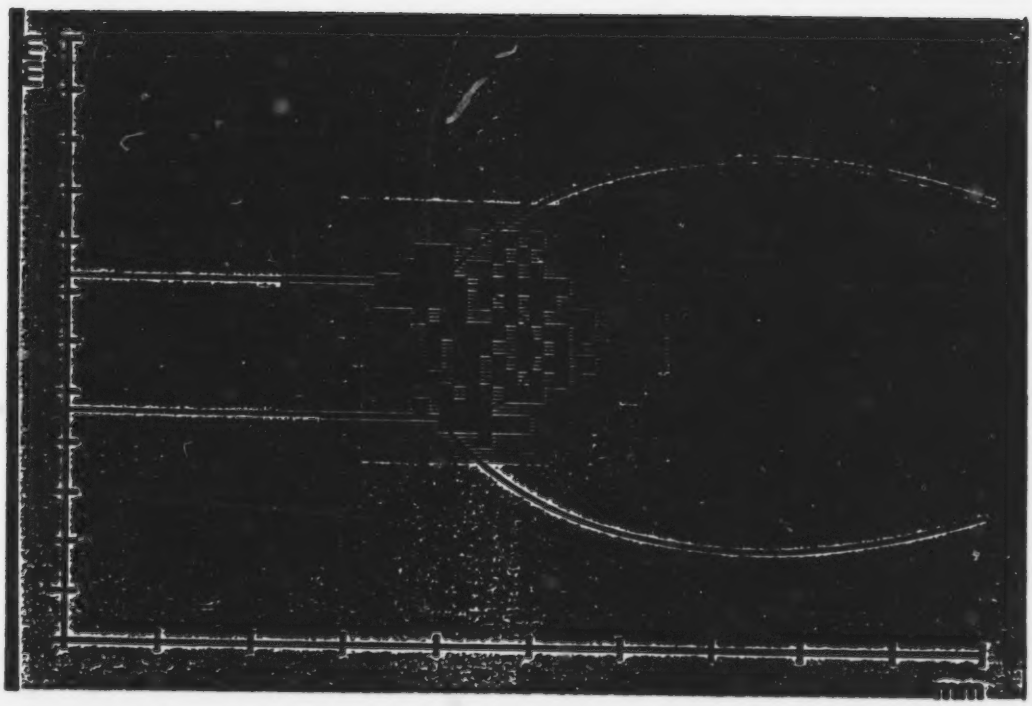


FIGURE 4.37: ESSAI 053
(JOINT BOUT A BOUT A BORDS DROITS)

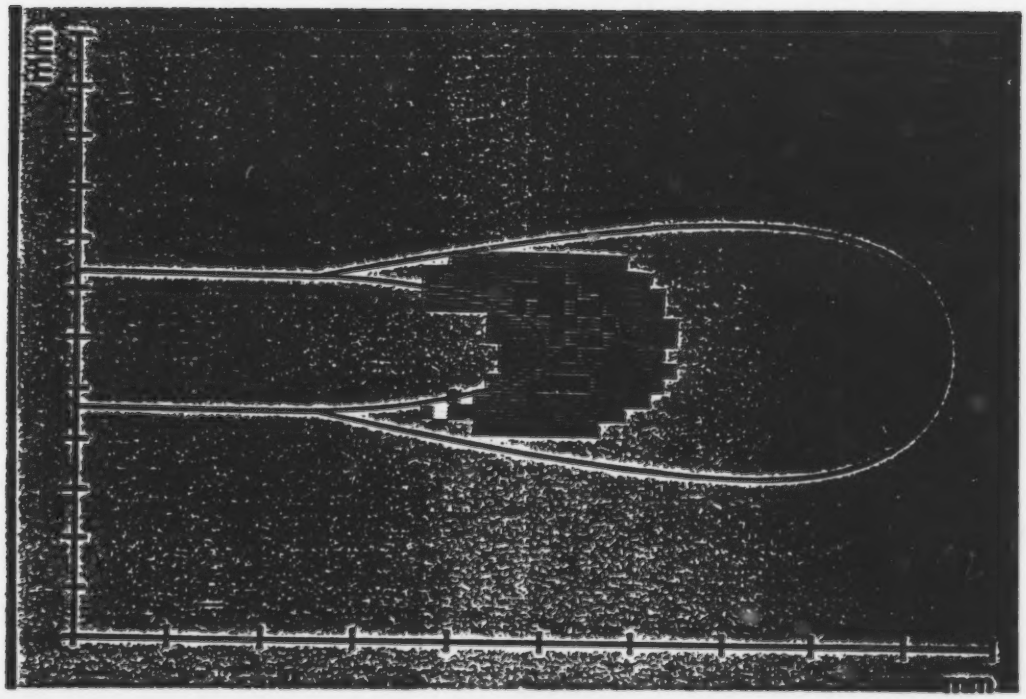


FIGURE 4.38: ESSAI 054
(JOINT BOUT A BOUT A BORDS DROITS)

## 4.3 Essais sur un joint bout à bout avec préparation en V (utilisation d'une fente).

Cette section regroupe les essais 11a à 11h inclusivement. L'utilisation d'une fente permet de restreindre le champ de vision et de détecter une ligne transversale au joint. L'essai 11a (figure 4.13 ) est une acquisition prise directement à travers le bain de fusion. Aucun joint n'est visible. L'essai 11b (figure 4.14 ) est pris 0.7 mm. en avant du bain. Le bain présentant des cornes de dimensions 0.8 mm., le signal est donc directement au travers des cornes. Le joint est bien visible. Les essais 11c (figure 4.15 ) 11d (figure 4.16 ) 11e (figure 4.17 ) 11f (figure 4.18 ) 11g (figure 4.19 ) donnent respectivement les signaux obtenus à des distances variant de 1.1 à 3.5 mm. en avant du bain de fusion. L'essai 11h (figure 4.20) donne un signal à la limite de détection i.e 4 mm. en avant du bain.

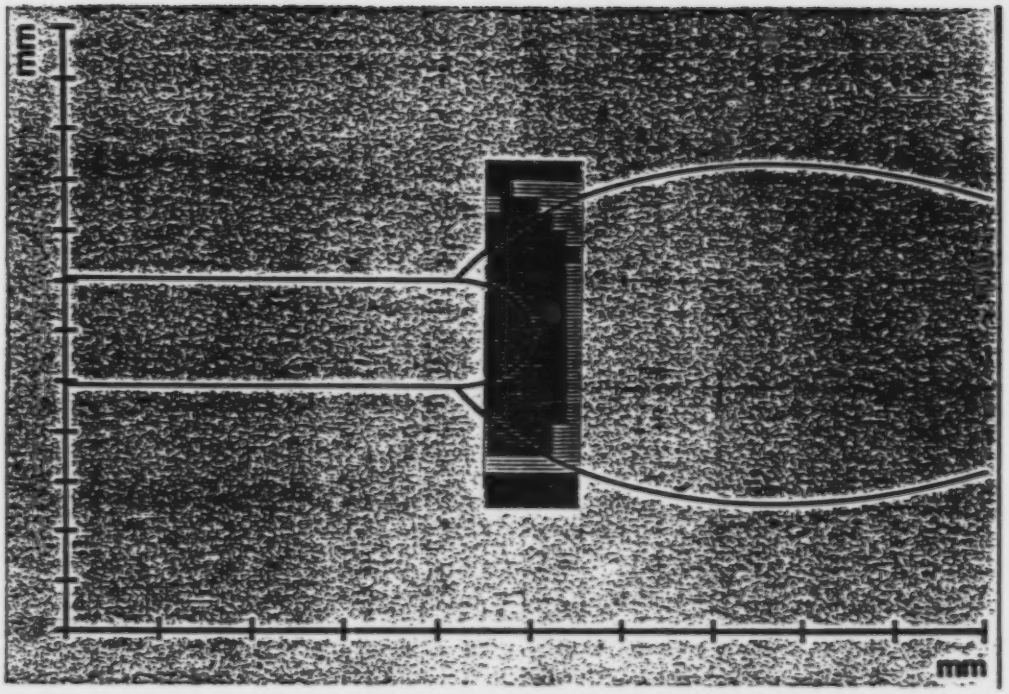



FIGURE 4.13: ESSAI 11A
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

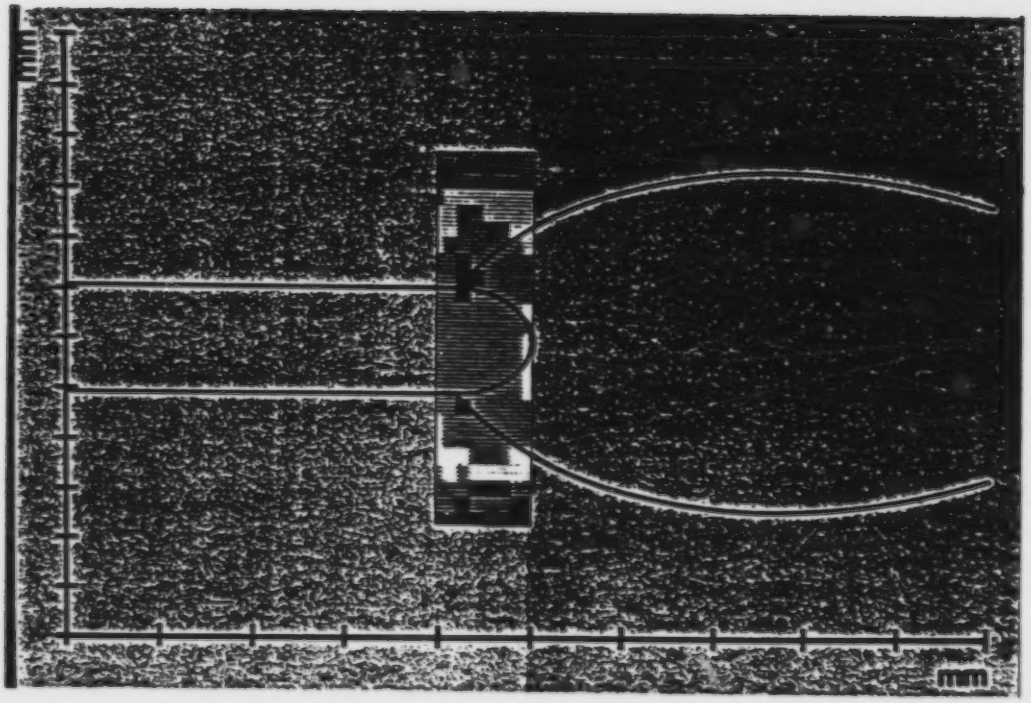



FIGURE 4.14: ESSAI 11B
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

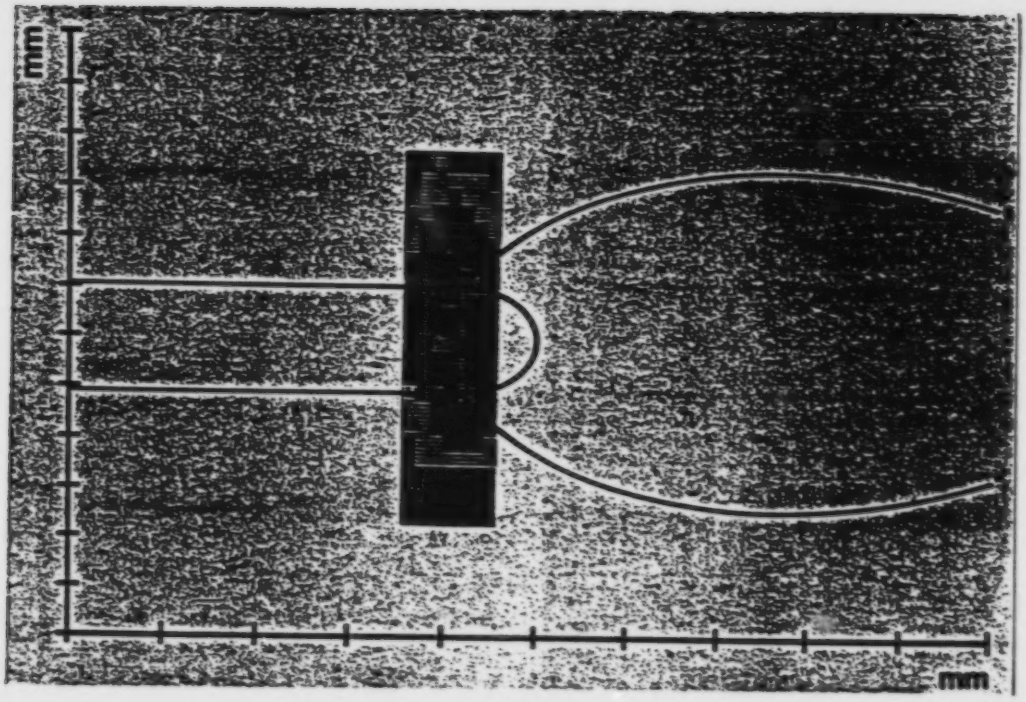



FIGURE 4.15: ESSAI 11C
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

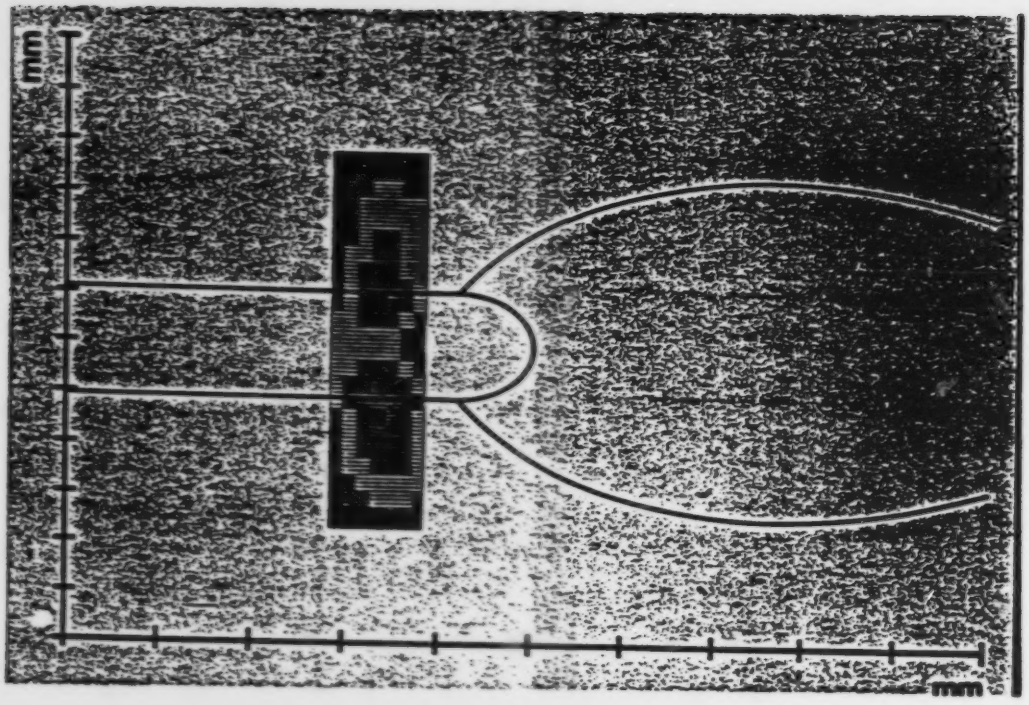



FIGURE 4.16: ESSAI 11D
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

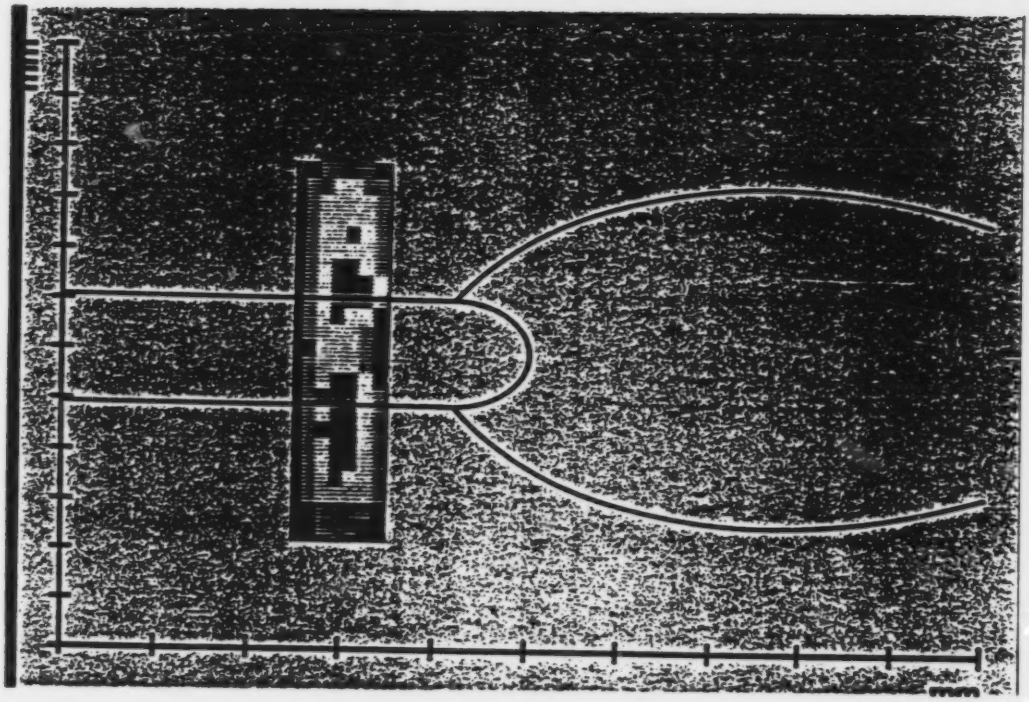


FIGURE 4.17: ESSAI 11E
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

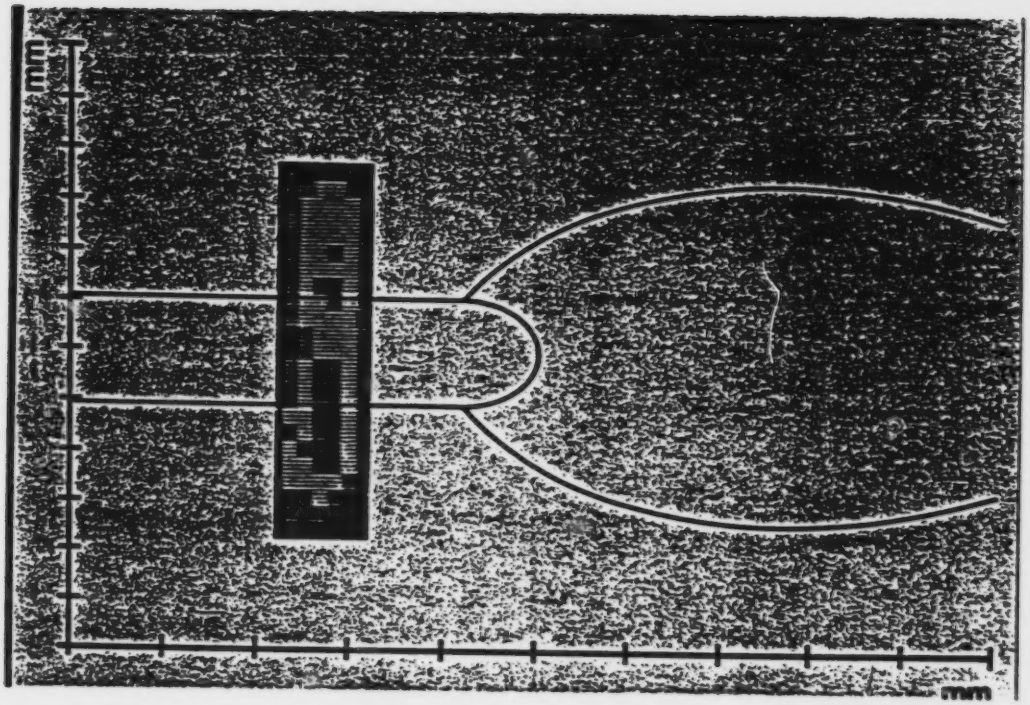



FIGURE 4.18: ESSAI 11F
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

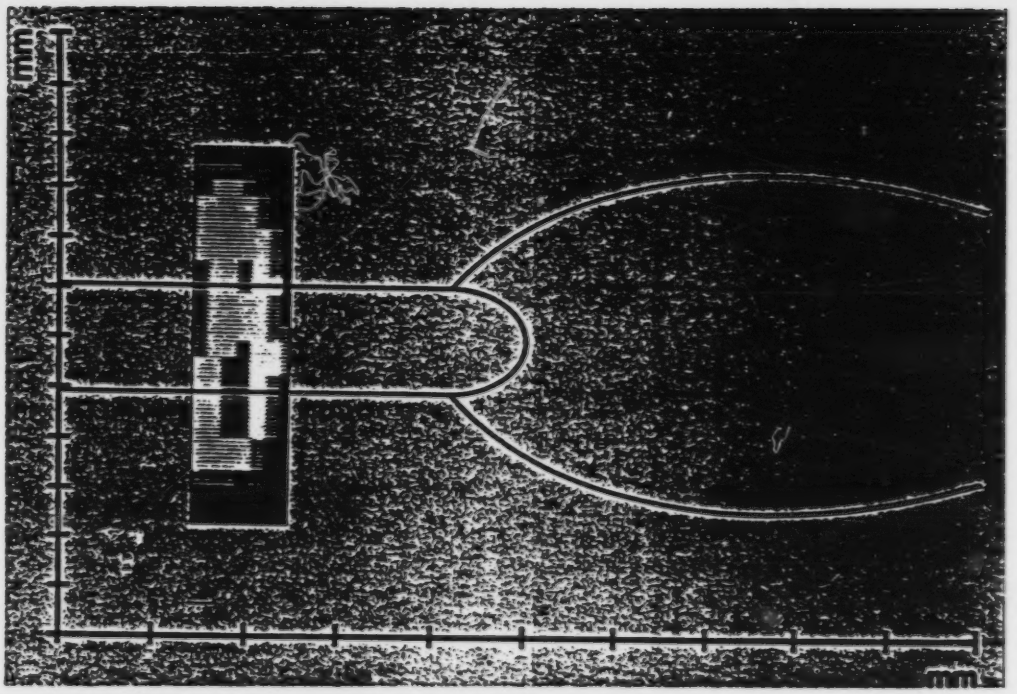



FIGURE 4.19: ESSAI 11G
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

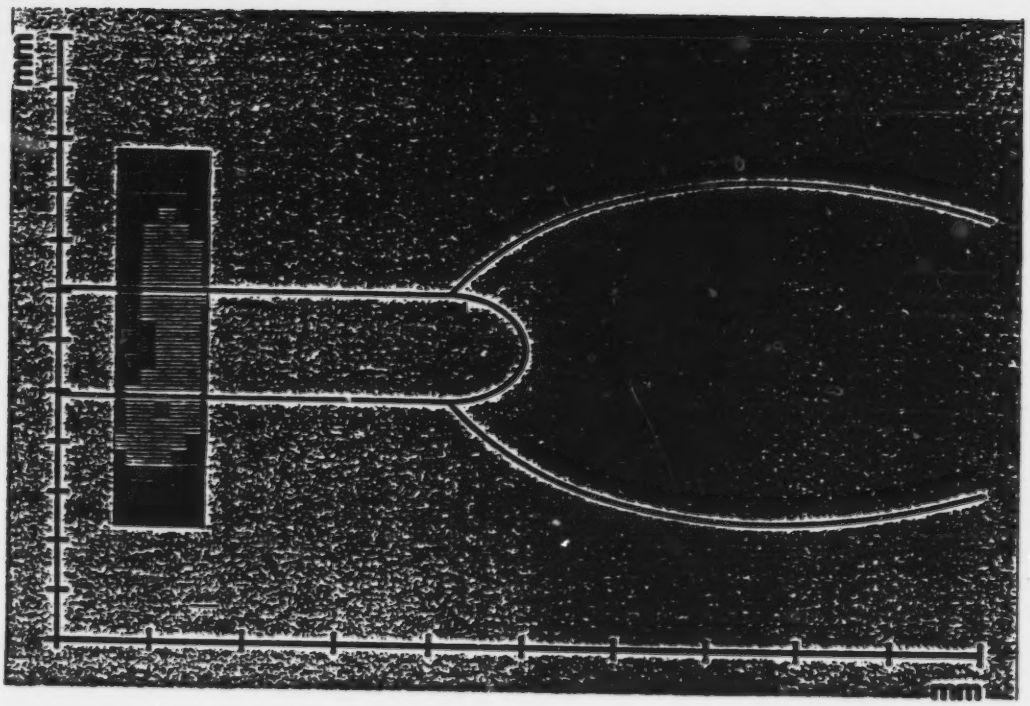


FIGURE 4.20: ESSAI 11H
(JOINT BOUT A BOUT AVEC PREPARATION EN V)

## 4.4 Essais sur un joint bout à bout à bord droit (utilisation d'une fente).

Cette section regroupe les essais 017 à 024. Les essais 017 (figure 4.26) et 018 (figure 4.27) sont obtenus par lecture directe à travers le bain de fusion. Aucun joint n'est visible. Les essais 019 (figure 4.28) et 020 (figure 4.29) sont pris à 0.4 mm. et 0.7 mm. en avant du bain de fusion. Comme les cornes ont près d'un mm., le signal est obtenu directement au travers des cornes du bain. L'essai 021 (figure 4.30) est pris à une distance de 1.4 mm. Le joint est facilement identifiable et les cornes du bain sont à l'extérieur de la fenêtre de détection. Les essais 022 (figure 4.31), 023 (figure 4.32) et 024 (figure 4.33) montrent respectivement les images obtenues à des distances allant de 1.7 mm. à 3 mm. en avant du bain de fusion. Le joint est identifiable dans chacun des cas.

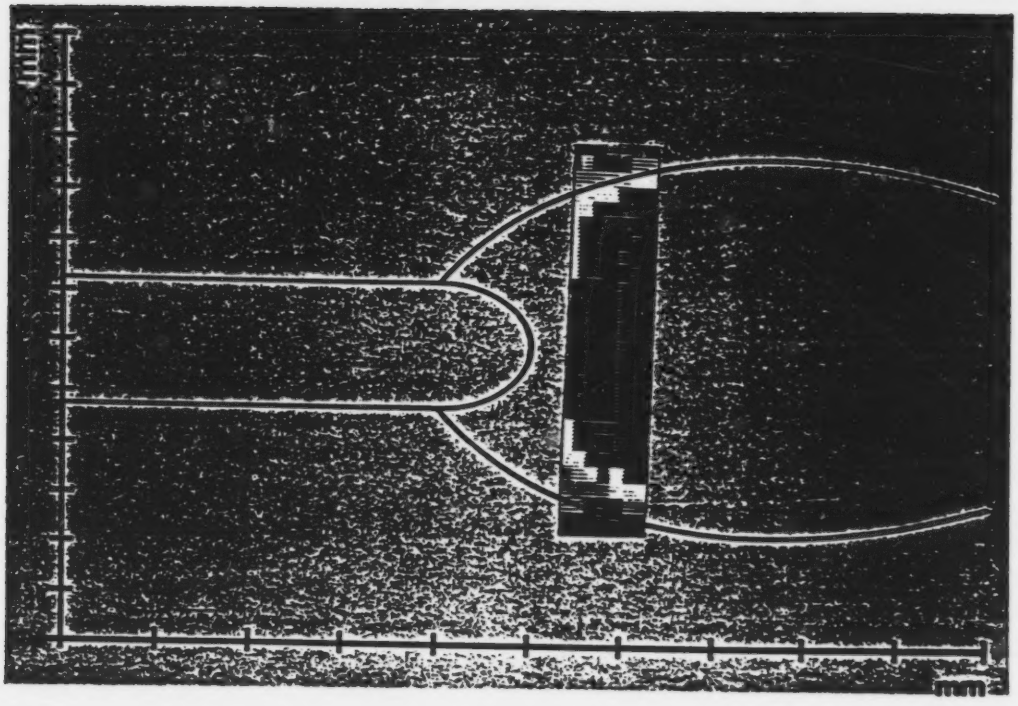


FIGURE 4.26: ESSAI 017
(JOINT BOUT A BOUT A BORDS DROITS)

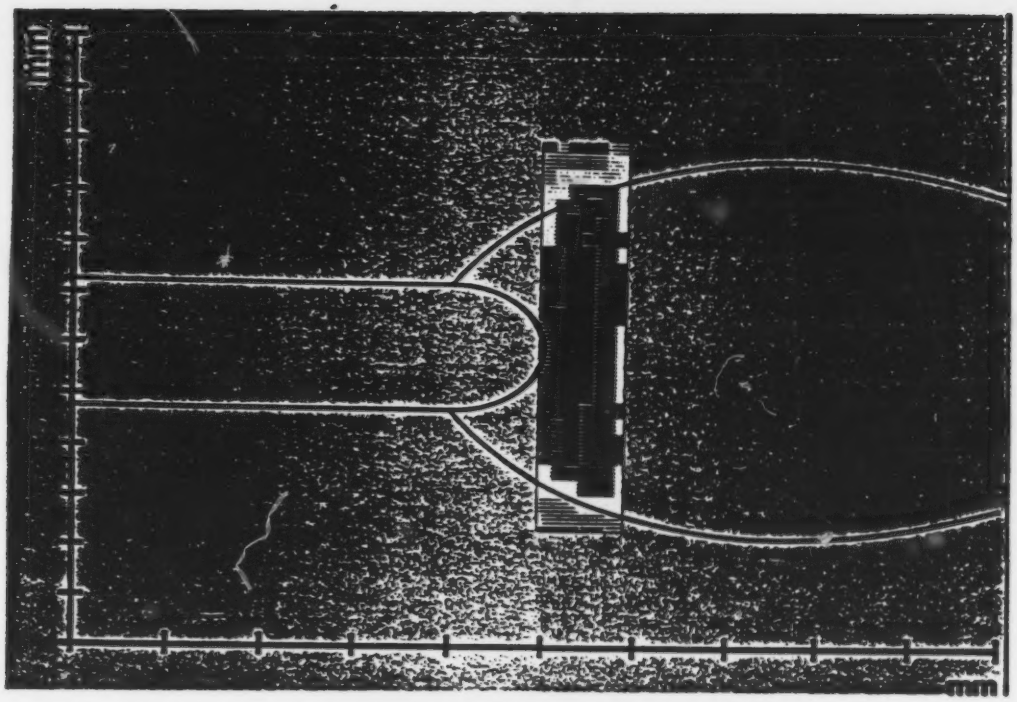


FIGURE 4.27: ESSAI 018
(JOINT BOUT A BOUT A BORDS DROITS)

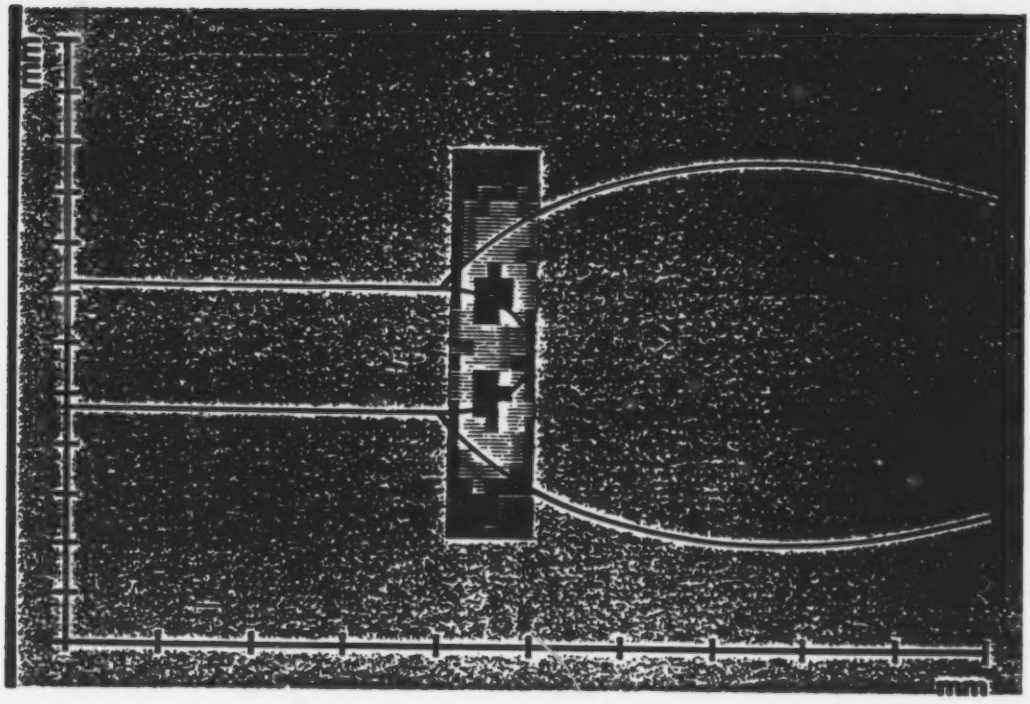


FIGURE 4.28: ESSAI 019
(JOINT BOUT A BOUT A BORDS DROITS)

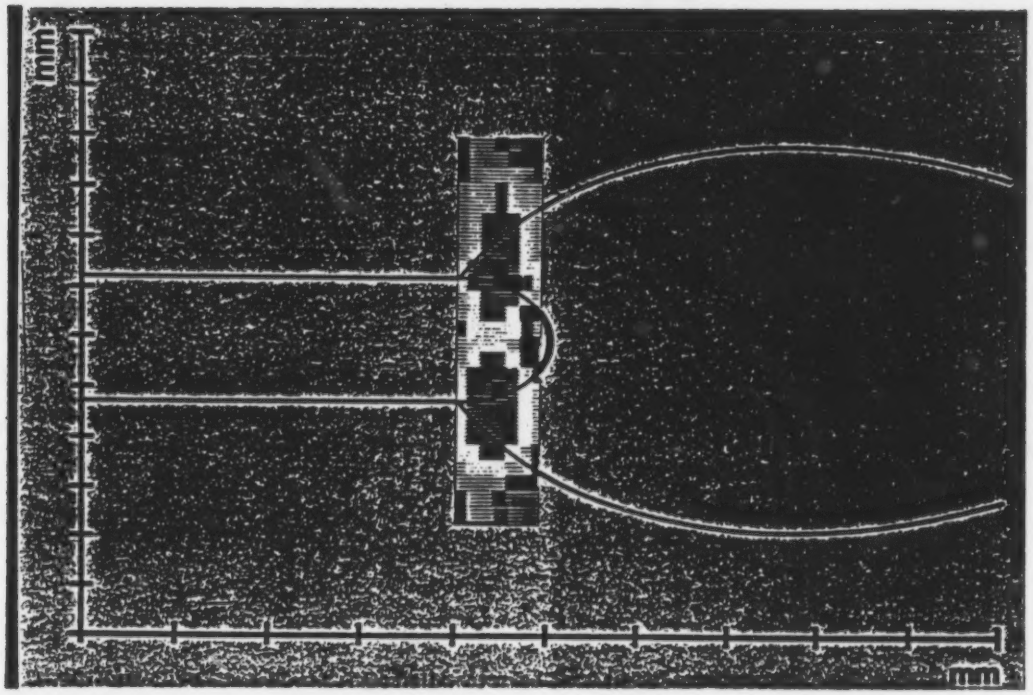



FIGURE 4.29: ESSAI 020
(JOINT BOUT A BOUT A BORDS DROITS)

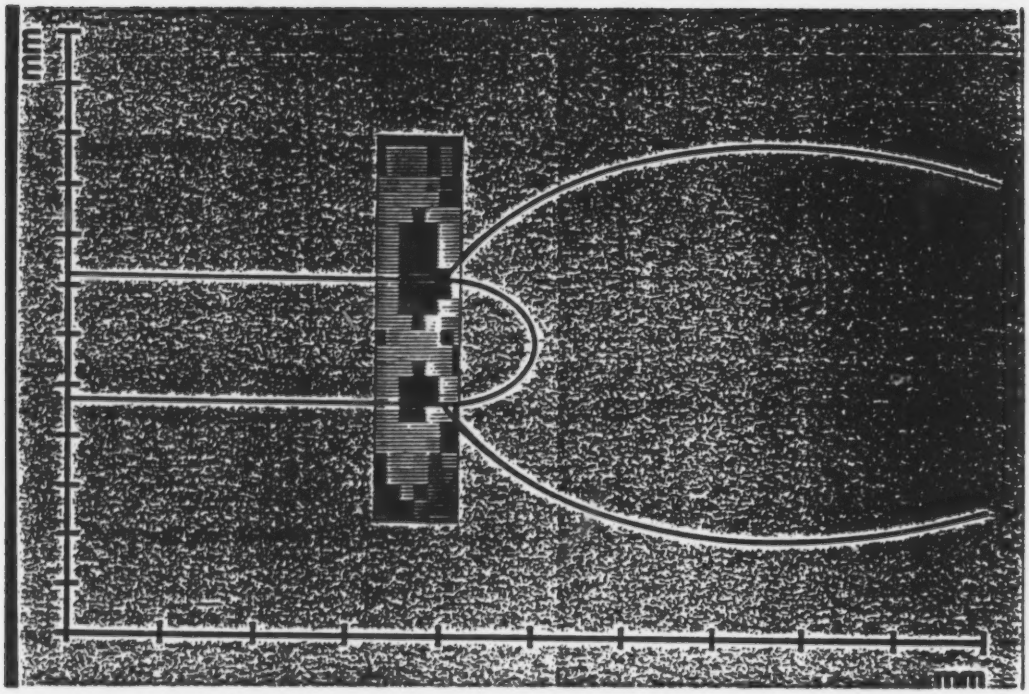



FIGURE 4.30: ESSAI 021

(JOINT BOUT A BOUT A BORDS DROITS)

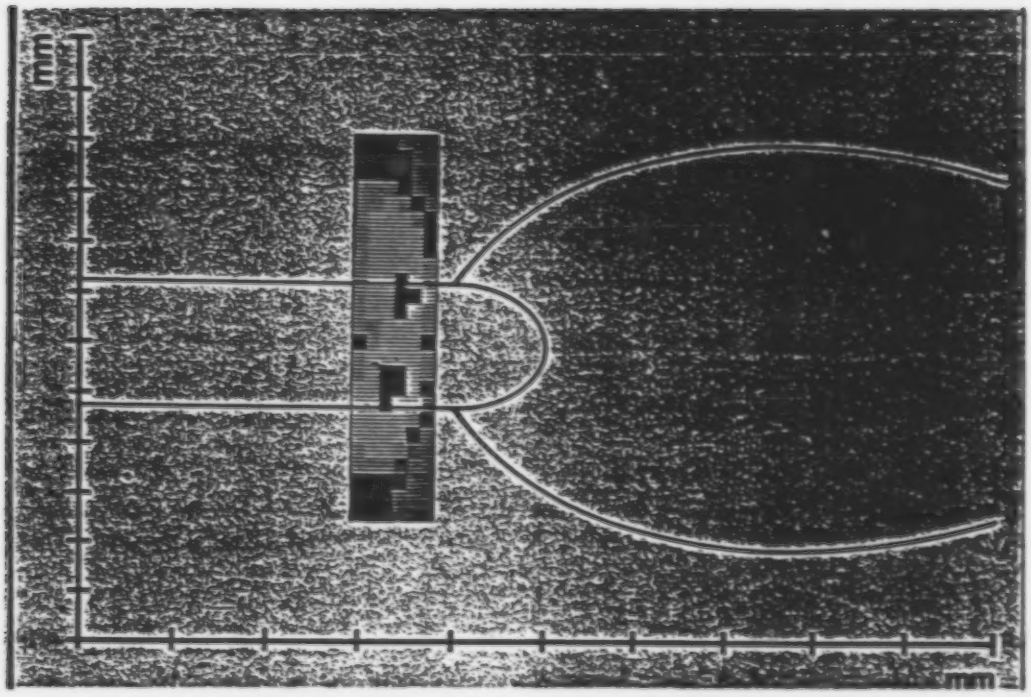



FIGURE 4.31: ESSAI 022
(JOINT BOUT A BOUT A BORDS DROITS)

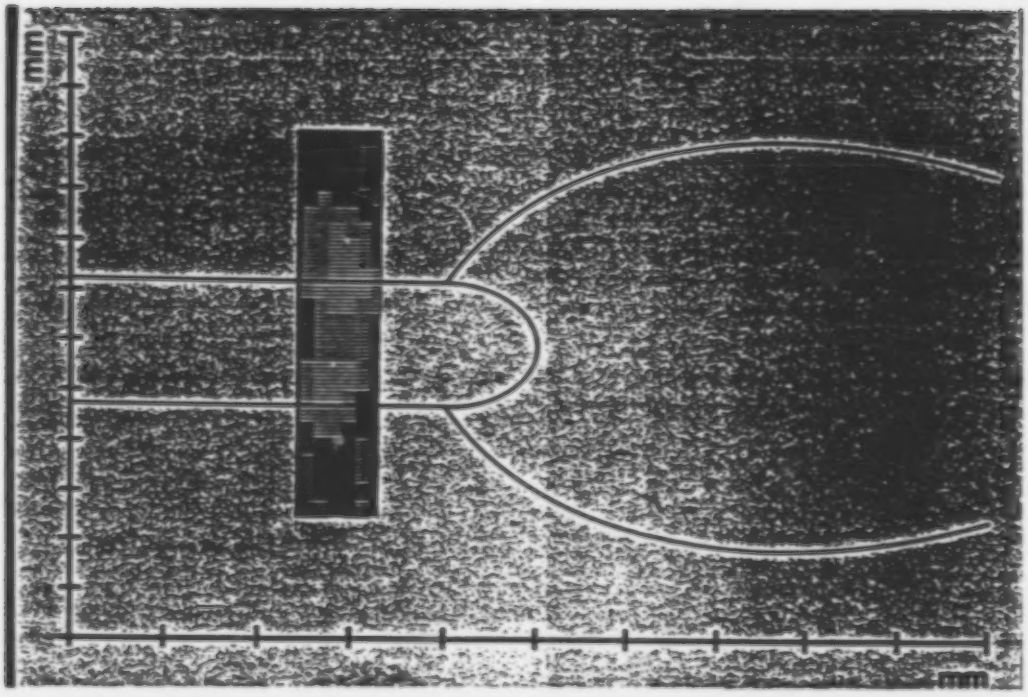



FIGURE 4.32: ESSAI 023
(JOINT BOUT A BOUT A BORDS DROITS)

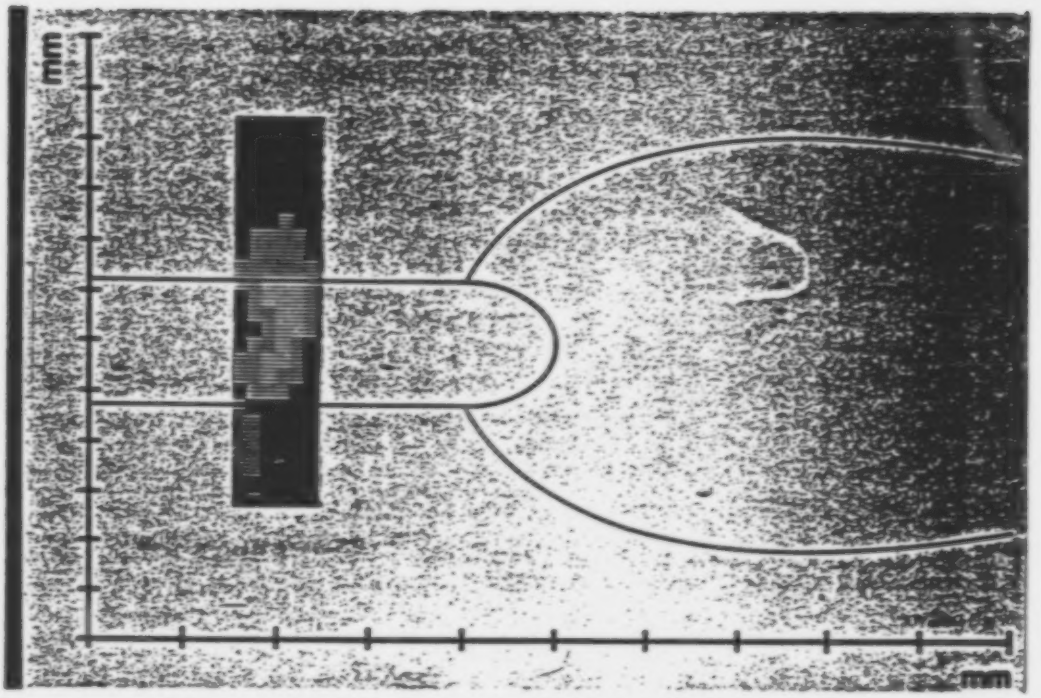



FIGURE 4.33: ESSAI 024
(JOINT BOUT A BOUT A BORDS DROITS)

## 4.5 Essais sur un joint à recouvrement et sur un joint d'angle (pleine image).

Cette section regroupe les essais 043 et 045. L'essai 043 (figure 4.39) représente un joint d'angle soudé à plat. Le bain de fusion a une forme arrondie et le signal ne permet pas de détecter le joint. L'essai 045 (figure 4.40) représente un joint à recouvrement. Dans ce cas-ci le joint n'est pas visible non plus. Le bain a une forme arrondie et ne présente pas d'effet de cornes. En conclusion, pour les joints d'angle et les joint à recouvrement, les images obtenues ne permettent pas de distinguer l'emplacement du joint à souder.

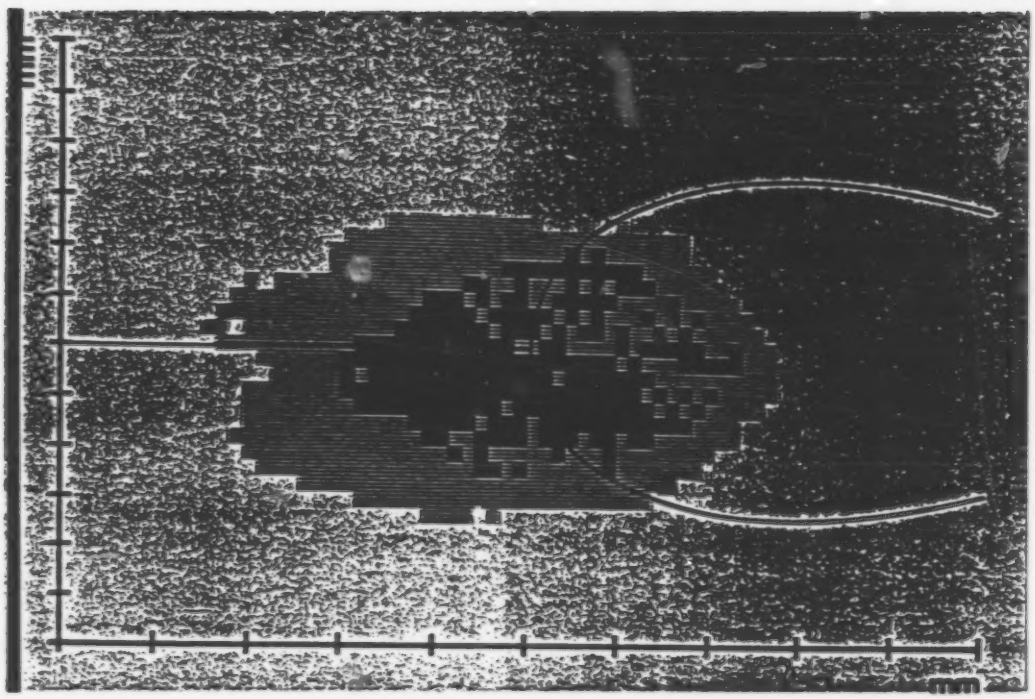


FIGURE 4.39: ESSAI 043
(JOINT BOUT A BOUT A ANGLE)

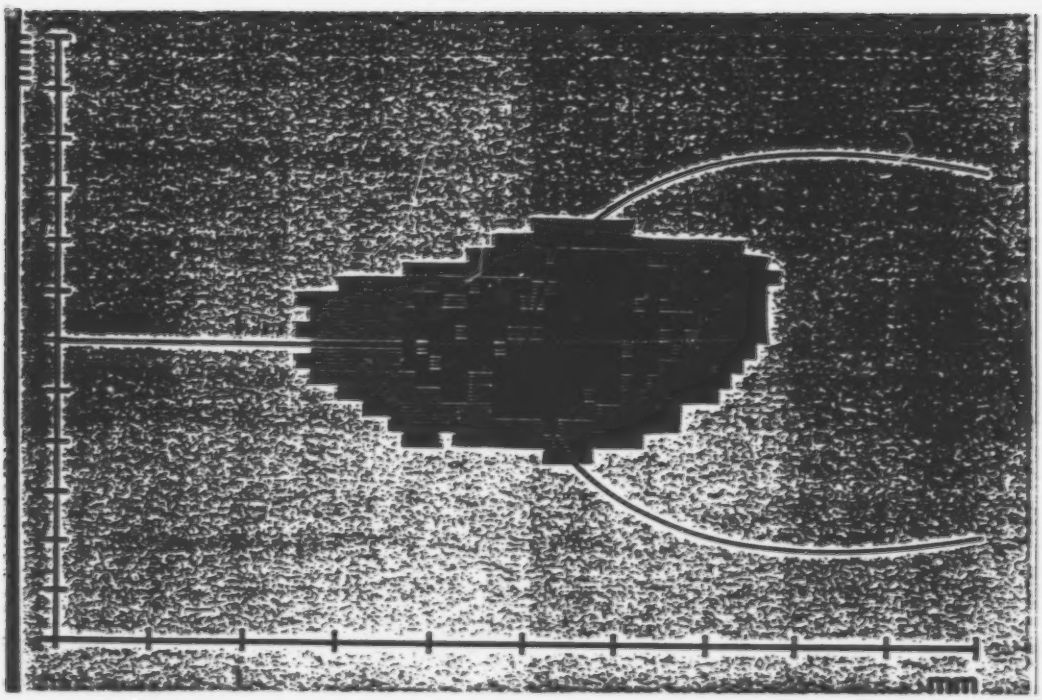


FIGURE 4.40: ESSAI 045
(JOINT BOUT A BOUT A RECOUVREMENT)

## 4.6 Essais de suivi de joint.

Le logiciel de suivi de joint est expliqué en détail à la section 3 du présent rapport. L'image obtenue à l'aide d'une fente de 1 mm. de largeur permet de restreindre le champ de vision à une colonne transversale au joint à souder. En prenant une colonne de référence non exposée au signal infra-rouge on débarasse le signal à traiter de l'effet de dérive des capteurs. La figure 4.41 montre 2 profils de température obtenus en échantillonnant une colonne transversale à un joint bout à bout à bord droit.

Le programme SUIVI.C permet de visualiser le profil d'une colonne en même temps que s'effectue le contrôle de suivi de joint. Les profils montrés sont une copie sur imprimante de l'image obtenue sur l'écran couleur de l'ordinateur IBM PC. Sur le coté droit de l'écran, on peut voir les différents paramètres correspondants à l'état actuel du système. En autres, on peut observer la position des deux sommets du profil (Sommet_0 et Sommet_1) ainsi que la position du centre du joint (Position). Les paramètres Czero et Cprofil correspondent respectivement à la colonne de référence du zéro et à la colonne du profil absolu.

La différence entre ces deux colonnes donne le profil relatif servant au contrôle de suivi de joint. Il est à remarquer que le programme utilise 21 des 48 capteurs disponibles de la caméra. Le paramètre ligne Bas donne le numéro du premier senseur utilisé. Le paramètre Statut_P nous indique si le profil analysé est valide. Dans le cas où on ne réussi pas à extraire deux et seulement deux sommets dans le profil (voir profil # 91), le statut devient égal à zéro. Ce statut est ensuite utilisé dans la stratégie de commande.

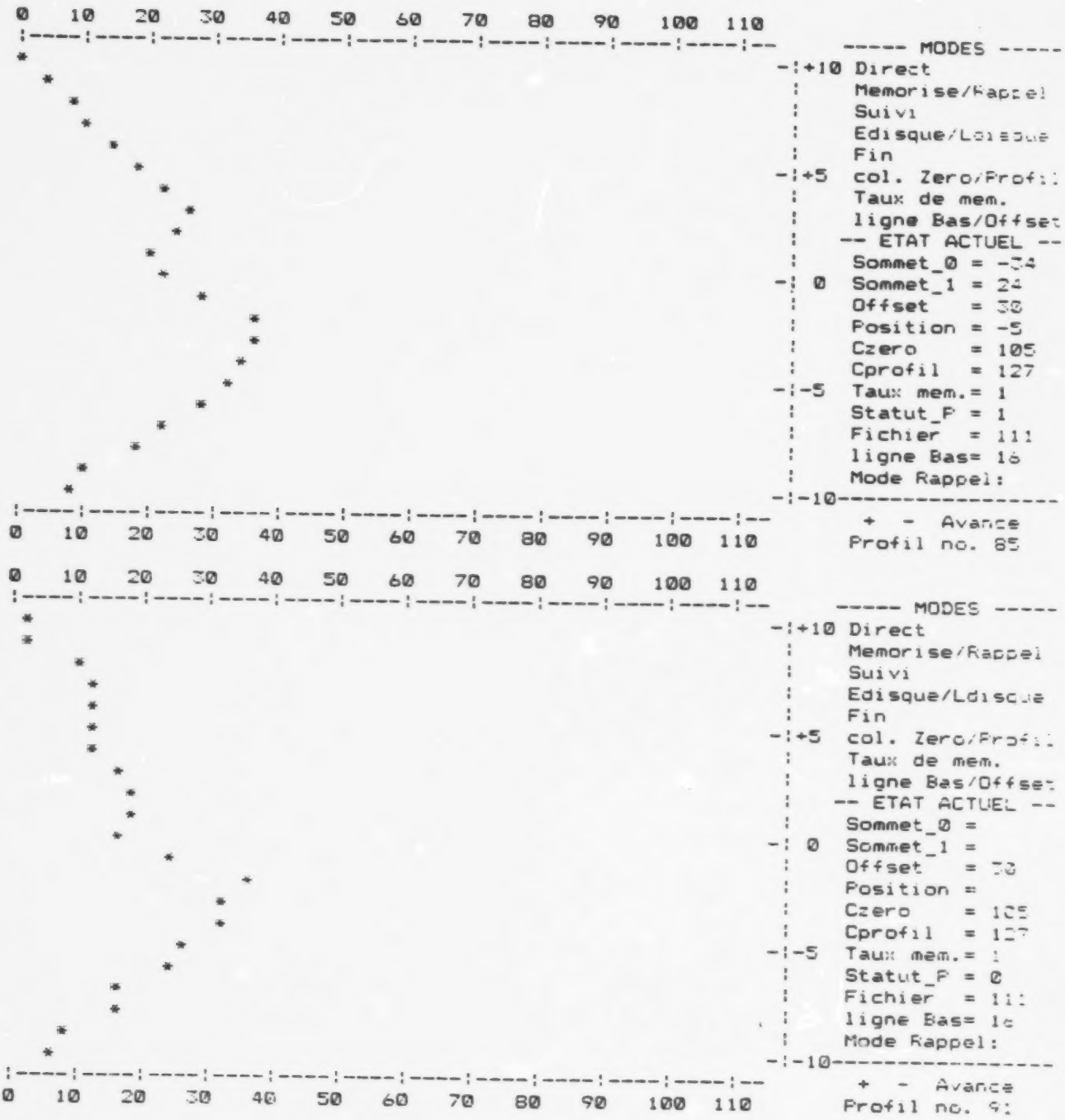



Figure 4.41: PROFILS POUR UN JOINT BOUT A BOUT A BORD DROIT

La figure 4.42 montre aussi un profil de température représentant un joint bout à bout avec préparation en V. On peut remarquer la différence de forme avec le profil précédent. Le joint avec préparation en V présente un profil plus accentué avec une distance sommet à sommet plus grande. On peut donc plus facilement faire le suivi pour ce type de joint. La figure 4.43 montre une plaque présentant un joint bout à bout avec préparation en V avant et après soudage. L'écartement est de 2.5 mm. et l'angle d'inclinaison par rapport à la direction de soudage est de cinq degrés. Pour des angles supérieurs à cinq degrés le cordon de soudure a tendance à suivre le côté extérieur du coupon.

Un coupon présentant un joint bout à bout à bords droit est montré à la figure 4.44 . Dans ce cas-ci la préparation a été effectuée à l'aide d'une torche de coupage oxyacétylénique. Les bords sont très irréguliers et présentent des courbures très aigus. Le suivi est quand même effectué précisément et sans bavures. Le suiveur développé est donc fiable et utilisable pour les joints bout à bout à bords droits et avec préparation en V. Les limitations du système ainsi que l'évalation et les recommendations sont données à la conclusion de ce rapport.

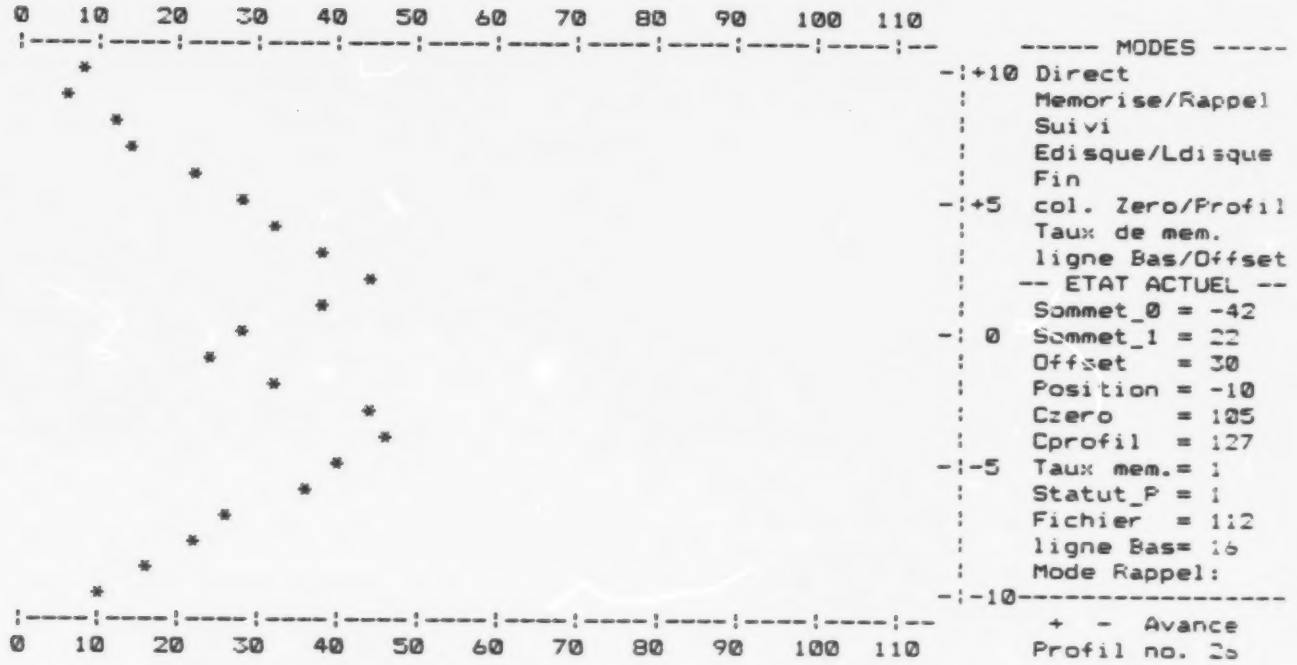



Figure 4.42: PROFIL POUR UN JOINT BOUT A BOUT AVEC PREPARATION EN V

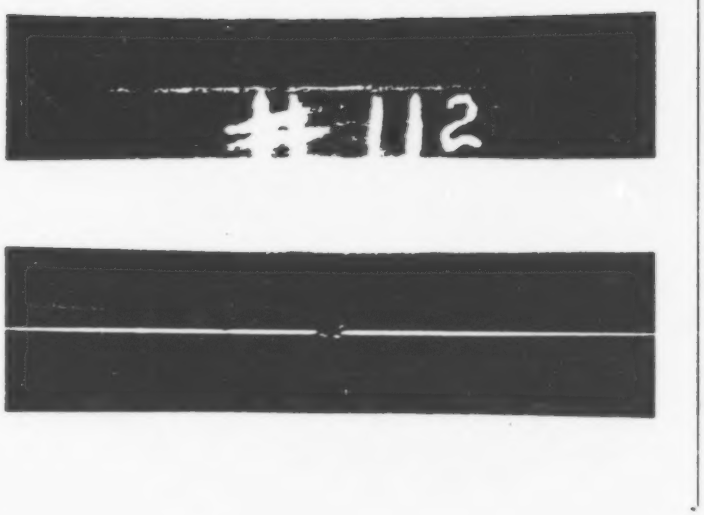


FIGURE 4.43: COUPON D'ESSAI

PLAQUE DROITE AVEC PRERARATION EN V ET INCLINAISON DE CINQ DEGRES

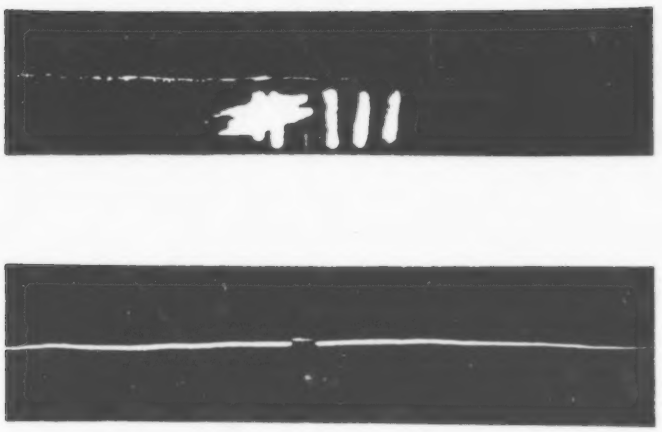


FIGURE 4.44: COUPON D'ESSAI
PLAQUE CURVILIGNE AVEC PREPARATION A BORDS DROITS

## 5. CONCLUSION

Un nouveau capteur infra-rouge de type industriel basé sur une caméra AGA THERMOVISION modèle 110 a été développé et intégré à une armoire de contrôle pour permettre le suivi de joints bout à bout à bords droits. Du même coup un système complet d'imagerie infra-rouge a été développé pour permettre l'acquisition d'images du bain de fusion et de la zone en périphérie du bain.

Un logiciel de suivi de joint utilisant les résultats de recherche obtenus lors des contrats CNRC OSD 82-00041 et CNRC OSD83-00035 a été implanté sur IBM PC.

Les essais de suivi ont été effectués à l'aide du logiciel SUIVI.C présenté à la section 3. La fréquence de contrôle est de 5 Hz. Vingt et un capteurs sont utilisés pour couvrir un champ de vision circulaire de 7.5 mm. de diamètre. La résolution d'un capteur est donc de 350 microns.

Le système a été évalué pour une application de suivi de joints bout à bout à bord droits et de joints bout à bout avec préparation en V.

Joints bout à bout avec préparation en V.:

Un écartement des plaques supérieur à 0.8 mm. est nécessaire pour détecter le joint. Des essais avec écartement allant jusqu'à 2.8 mm. ont été effectués. Dans chacun de ces cas le joint est facilement identifiable (voir section 4.1). La région optimale pour la détection du signal de suivi est 3 mm. en avant du bain de fusion, bien que les essais d'imagerie ont montré qu'il était possible de repérer le joint jusqu'à 7 mm. en avant du bain de fusion.

Joints bout à bout à bords droits:

Dans le cas des joints bout à bout à bord droits, des écartements allant de 0,7 mm. à 3 mm. peuvent être détectés de façon fiable. La région optimale de détection se situe de 1.5 à 2 mm. en avant du bain de fusion et le maximum, dans ce cas serait de 4 mm.

Joints d'angle et à recouvrement:

Des essais ont aussi été effectués sur des joints d'angle et des joints à recouvrement. Dans ces deux cas la forme arrondie du bain donne des isothermes sans brisure. Il n'est donc pas possible de discerner le joint.

En conclusion, le système de suivi de joint basé sur le nouveau capteur infra-rouge permet d'effectuer de façon fiable le suivi de joints bout à bout à bords droits et avec préparation en V. Ce nouveau suiveur corrobore l'hypothèse de base spécifiant qu'un système de détection infra-rouge sensible aux longueurs d'ondes plus grandes permette une détection des isothermes à une distance plus grande en avant du bain de fusion.

Le suiveur de seconde génération (1) était limité à une détection inférieure à 2 microns et de ce fait obligeait l'acquisition de profils de température obtenus par focalisation directe au niveau des cornes du bain de fusion. Cette limitation amenait toute sorte de problèmes dus aux mouvements du bain de fusion et provoquait la perte du joint de temps à autre. L'extension de la plage de détection à des longueurs d'ondes de 5 microns permet donc de s'affranchir de cette contrainte.

(1) contrat CNRC OSD 82-00041 et CNRC OSD83-00035

Limitations du suiveur de troisième génération:

A- Montage optique

Le système optique ayant été conçu originalement pour utiliser un guide de fibres optiques, les modifications nécessaires pour utilisation en focalisation directe ont amené certaines restrictions. Une de ces limitations est la restriction du champ de vision à 7.5 mm. de diamètre. Seulement 21 des capteurs sont donc utilisé sur une possibilité de 48. De plus la philosophie première fût de ne pas modifier de façon permanente la caméra AGA Thermovision 110, d'où certaines limitations de l'optique inhérentes à l'appareil commercial.

B- Volume:

Le volume utilisé par l'installation optique et la caméra est substantiel. Dans sa forme actuelle son emploi peut être viable dans une application dédié sur un banc de soudage par exemple. Le matériel actuel ne pourrait pas être utilisé sur un robot soudeur. Pour résoudre ce problème, il faudrait développer une méthode de fabrication de faisceau optique adaptée à l'extrême fragilité des fibres IR au fluor.

C-Sensibilité de la détection:

Les essais effectués ont permis de confirmer une grande dépendance des signaux infra-rouges vis à vis les mouvements du bain de fusion. En effet des changements de paramètres comme la vitesse de fil ou la vitesse de soudage ont un effet marqué sur la forme du bain de fusion (plus spécialement la forme des cornes du bain de fusion). Ces changements se retrouvent inévitablelemt dans les profils de température détectées.

Recommandations:

Nous croyons que de nouveaux développements sur le suiveur IR ne devraient être entrepris qu'avec la collaboration d'un partenaire industriel et pour une application dédiée. Les travaux à effectuer se feraient sous trois volets:

1- Modification de la caméra AGA Thermovision.

L'optique standard de la caméra serait remplacée par une optique appropriée pour l'application choisie. Si l'application requiert un robot soudeur, le système pourrait être miniaturisé en construisant l'électronique nécessaire autour du capteur et en relocalisant le tout dans un boîtier plus restreint. Une autre possibilité s'offrant à l'usager consisterait en l'utilisation d'un guide de fibres optiques infra-rouge peu encombrant au niveau de l'outil du robot. La caméra pourrait être disposée sur le dessus du bras du robot où l'encombrement n'est pas critique. Ceci nécessite évidemment des progrès substantiels du côté de la fabrication de fibres IR robustes.

2- Optimisation du logiciel de suivi.

Le logiciel devrait être optimisé pour travailler à une fréquence de 15 Hz de façon à égaler la performance de la caméra AGA. De plus une procédure de passage de points de soudure devrait être développée pour assurer la fiabilité de l'opération.

3- Intégration de l'ensemble complet:

Des travaux d'intégration seraient nécessaires pour assurer une homogénéité des composantes matérielles et logicielles.

# ANNEXE A

## Logiciel PSEUDO.C

```
/* 30 avril 1986 */
/*****************************************************************************

                                PSEUDO.C

*****************************************************************************

    Version 1.0
(c) Tous droits reserves,
    Yves Chartier, Institut de Soudage du Canada, 1986

*****************************************************************************/

#include <stdio.h>

#define ALPHA_16                    9
#define BORDER_COLOR                0
#define COLOR_SELECT_REGISTER    0X3D9
#define CONTROL_SEGMENT         0X8800
#define CRF                         1
#define DATA_REGISTER            0X3D5
#define FIN_CYCLE                   3
#define INDEX_REGISTER           0X3D4
#define LIGNE25_BASE             0XF00
#define MODE_RFD                    0
#define MODE_SELECT_REGISTER     0X3D8
#define RCT                         8
#define R0                          0
#define R1                          1
#define RAM_SEGMENT             0X8C00
#define RAZ                         6
#define STATUS_REGISTER          0X3DA
#define SUP_MASK                  0XF0
#define TAMPON_SEGMENT          0X5000
#define VERTICAL_MASK               8
#define VIDEO_DISABLE               0
#define VIDEO_PAGE1_SEGMENT     0XB900

/*
```

```
*/

char image_brute[3840];
char image_alpha[3840];
char palette_attribut[255];
char *line_pointer_array[24];
int _fmode = 0X0000;

main()
{
   int commande_sortie;

   depart();

   commande_sortie = 0;
   while (!commande_sortie) {
      if (kbhit()) {
         switch (getch()) {
         case 'd': case 'D':mode_direct(); break;
         case 's': case 'S':mode_sauve(); break;
         case 'r': case 'R':mode_rappel(); break;
         case 'e': case 'E':mode_ecriture_disque(); break;
         case 'l': case 'L':mode_lecture_disque(); break;
         case 'f': case 'F':commande_sortie = 1; break;
         default: break;
         }
      }

   }

   retour();
}

/*
```

```
*/
/*************************************************************************

                     acquisition_image()                    P-③

**************************************************************************

Cette routine fait l'acquisition d'une image de 48 lignes par 80 colonnes.
Par rapport a l'image emmagasinee dans la memoire de la carte AI, on fait
l'acquisition d'une colonne sur trois, soit de la colonne #9 a la colonne
#246.  Sur la carte AI, une colonne occupe 64 octets meme si on utilise
seulement 48 octets.  L'increment d'octets pour trois colonnes est donc
de 192 octets.  Le tampon image_brute[] contient 80 colonnes de 48 elements,
l'increment de son pointeur est donc de 48.

*************************************************************************/

acquisition_image()
{
    extern char image_brute[];
    char *pimage;
    int i, debut_colonne;

    pimage = image_brute;
    debut_colonne = 512;        /* 1er element de la colonne #9 */
    i = 80;

    while (i) {
        peek(RAM_SEGMENT, debut_colonne, pimage, 48);
        pimage += 48;
        debut_colonne += 192;
        i--;
    }
    return(debut_colonne);
}

/*
```

```
*/
/*****************************************************************************

                        acquisition_80_colonne()

*****************************************************************************

Cette routine fait l'acquisition d'une image de 48 lignes par 80 colonnes.
Par rapport a l'image emmagasinee dans la memoire de la carte AI, on fait
l'acquisition de la colonne #88 a la colonne#167.  Sur la carte AI, une
colonne occupe 64 octets meme si on utilise seulement 48 octets.
L'increment d'octets est donc de 64 octets.  Le tampon image_brute[]
contient 80 colonnes de 48 elements, l'increment de son pointeur est donc
de 48.

*****************************************************************************/

acquisition_80_colonne()
{
   extern char image_brute[];
   char *pimage;
   int i, debut_colonne;

   pimage = image_brute;
   debut_colonne = 5632;       /* 1er element de la colonne #88 */
   i = 80;

   while (i) {
      peek(RAM_SEGMENT, debut_colonne, pimage, 48);
      pimage += 48;
      debut_colonne += 64;
      i--;
   }
   return(debut_colonne);
}

/*
```

```
*/
/*********************************************************************

                          affiche_image()

**********************************************************************

Cette routine fait la conversion entre le tampon d'entree image_brute[] et
le tampon de sortie image_alpha[] pour obtenir une image en pseudo-couleur
sur l'affichage video.

Le tampon image_alpha est constitue de 24 lignes de 160 elements, soit 80
paires (caracteres, attribut).

  TABLE    0                                              159   LIGNE

  LPA23    CACACACACACACACACA..........................CACACA      0
  LPA22    CACACACACACACACACA..........................CACACA      1
              .          .          .          .          .
              .          .          .          .          .
              .          .          .          .          .
              .          .          .          .          .
              .          .          .          .          .
  LPA0     CACACACACACACACACA..........................CACACA     23

  La table line_array_pointer[] contient une serie de 24 pointeurs qui se
  deplacent sur leur ligne respective.  Comme le tampon image_brute[]
  est constitue de colonnes,  la table de pointeurs  facilite la conversion
  de colonnes en lignes.  La progression de l'indice de line_pointer_array[]
  est a l'inverse de celle des lignes de l'affichage video.  Ceci est du
  a la convention pour definir l'origine de l'image.  Dans le cas de la
  carte AI, le point (0,0) est situe dans le coin sud-ouest alors que pour
  l'affichage video, le point (0,0) est situe dans le coin nord-ouest.

*********************************************************************/

affiche_image()
{
   extern char image_brute[];
   extern char image_alpha[];
   extern char *line_pointer_array[];
   char *palpha, *(*pline), *pbrute;
   int i, j, statut0, statut1;

   /* Initialisation de line_pointer_array[] */
   palpha = image_alpha + 1;
   pline = line_pointer_array;
   for (i=0; i<24; i++) {
      *pline++ = palpha + (23-i) * 160;
   }
   /* Conversion de image_brute[] a image_alpha[] */
   pline = line_pointer_array;
   pbrute = image_brute;
   for (j=0; j<80; j++) {
      pline = line_pointer_array;
      for (i=0; i<24; i++) {
         *(*pline) =
            palette_attribut[(*pbrute >> 4) | (*(pbrute + 1) & SUP_MASK)];
         pbrute += 2;
         *pline++ += 2;
```

```
        }
    }
    /* Transfert de image_alpha[] vers la memoire de la carte video */
    palpha = image_alpha;
    for (i=0; i<8; i++) {
        statut0 = 0;
        statut1 = 0;
        /* Detection du video vertical enable: transition de 0 a 1 */
        while (!(!statut0 && statut1)) {
            statut0 = inp(STATUS_REGISTER) & VERTICAL_MASK;
            statut1 = inp(STATUS_REGISTER) & VERTICAL_MASK;
        }
        poke(VIDEO_PAGE1_SEGMENT, i * 480, palpha, 480);
        palpha += 480;
    }
    return;
}
/*
```

```
*/
/***********************************************************************
                         avance(compteur)
***********************************************************************
Cette routine fait defiler successivement les images en mode rappel.
***********************************************************************/

avance(compteur)

int compteur;

{
   int i;

   while (!kbhit()) {
      compteur++;
      if (compteur > 29) {
         compteur = 0;
      }
      ecriture_numero_image(compteur);
      rappel_image(compteur);
      affiche_image();
   }
   return(compteur);
}
/*
```

P- 9

```
*/
/*************************************************************************

                         decremente(compteur)                    P-(8)

**************************************************************************

Cette routine decremente le compteur de profil en mode rappel.

*************************************************************************/

decremente(compteur)

int compteur;

{
   compteur--;
   if (compteur < 000) {
      compteur = 29;
   }
   return(compteur);
}

/*
```

```
*/
/*****************************************************************************

                        demande_numero_fichier()                    D-9

*****************************************************************************

Cette routine dialogue avec l'operateur afin d'obtenir un numero de fichier
valide pour les routine de transfert vers le disque.

*****************************************************************************/

demande_numero_fichier()

{
   char entree_clavier[8];
   char *pclavier;
   char digit, valeur;
   int i, ligne25_offset, nombre_digit, numero;

   ecriture_question(2);

   valeur = 0;
   nombre_digit = 3;
   ligne25_offset = LIGNE25_BASE + 154;
   pclavier = entree_clavier;

   i = 3;
   while (i) {
      poke (VIDEO_PAGE1_SEGMENT, ligne25_offset, &valeur, 1);
      ligne25_offset += 2;
      i--;
      }
   ligne25_offset = LIGNE25_BASE + 154;

   while (nombre_digit) {
      if (kbhit()) {
         digit = getch();
         if (digit >= '0' && digit <= '9') {
            *pclavier = digit;
            poke (VIDEO_PAGE1_SEGMENT, ligne25_offset, pclavier, 1);
            nombre_digit--;
            ligne25_offset += 2;
            pclavier++;
         }
         if (digit == '\b') {
            return(-1);
         }
      }
   }
   *pclavier++ = '\0';
   stcd_i(entree_clavier, &numero);
   return(numero);
}

/*
```

```
*/
/*****************************************************************************

                              depart()

******************************************************************************

Cette routine de depart du programme PSEUDO.C procede a l'initialisation de
la carte d'affichage video en mode alphanumerique/couleur. On a ainsi un
espace de 25 x 80 caracteres avec attributs de couleur "foreground" et
"background".  En remplissant un espace de 24 x 80 par le caractere
graphique de demi-hauteur chr$(220), on arrive a une resolution de 48 x 80
en jouant sur les attributs de couleur. Le curseur servant au dialogue a
l'usager est place a la derniere ligne(#25).  Pour plus de details, le
lecteur peut consulter le manuel de reference technique de IBM pour la carte
d'affichage video alphanumerique/couleur.

La routine d'affichage des pseudo-couleurs utilise la page 1 de la carte
graphique (VIDEO_SEGMENT = 0XB900).  Les messages utilisant la fonction
printf() sont envoyes vers la page 0 puisse que cette fonction est intimement
reliee a l'environnement duPC DOS.  En activant la fonction ^P lors de
l'execution du programme, on peut alors avoir des messages a l'imprimante
tout en preservant l'ecran. Ces messages ne devront pas occuper plus que
20 lignes car le "software scrolling" du PC DOS active le bit "Enable Blink
Attribute".  Ceci a commme consequence d'enlever 8 couleurs pour l'attribut
"background".  Avant de debuter l'execution du programme, on doit envoyer
la commande CLS afin de replacer le curseur en haut de l'ecran pour ainsi
empecher le "soft scrolling".  Il est a noter que la fonction retour() a la
fin du programme PSEUDO.C replace l'affichage a la page zero afin de replacer
l'ecran dans l'environnement du PC DOS.

De plus, cette routine initialise la table palette_attribut[] a partir de
la table de pseudo-couleurs palette_couleur[].  Il est a noter que le
chr$(220) donne un caractere selon la convention suivante:

             ---
            ! B !    Cette convention est intimement reliee a la routine
             ---     affiche_image() au niveau de l'ordre dans lequel est
            ! F !    construit le tampon image_alpha[].
             ---

*****************************************************************************/

depart()
{
   extern char image_alpha[];
   extern char palette_attribut[];
   static char palette_couleur[16] =
      { 0, 0, 0, 0, 1, 1, 9, 9, 3, 2, 10, 14, 12, 4, 5, 5 };
   static int registre_6845[16] =
      { 0X71, 0X50, 0X5A, 0X0A, 0X1F, 0X06, 0X19, 0X1C,
        0X02, 0X07, 0X06, 0X07, 0X08, 0X00, 0X00, 0X80
      };
   static char ligne25_commande[80] =
      {"DIRECT SAUVE RAPPEL EDISQUE LDISQUE FIN"};
   static char ligne25_attribut[80] =
      {  31,  30,  30,  30,  30,  30,  30,  31,  30,  30,
         30,  30,  30,  31,  30,  30,  30,  30,  30,  30,
         31,  30,  30,  30,  30,  30,  30,  30,  31,  30,
         30,  30,  30,  30,  30,  30,  31,  30,  30,  30,
```

P-10

```
        30,  30,  30, 160, 160, 160, 160, 160, 160, 160,
       160, 160, 160, 160, 160, 160, 160, 160, 160, 160,
       160, 160, 160, 160, 160, 160, 160, 160, 160, 160,
       160, 160, 160, 160, 160, 160, 207, 207, 207, 207
    };
    int i, j, *pregistre;
    int ligne25_offset;
    char *pimage_alpha, *pligne25;

    /* Desactivation du signal video */
    outp(MODE_SELECT_REGISTER, VIDEO_DISABLE);
    /* Initialisation du 6845 pour le mode 25 x 80 couleur */
    pregistre = registre_6845;
    for (i=0; i<16; i++) {
       outp(INDEX_REGISTER, i);
       outp(DATA_REGISTER, *pregistre++);
    }
    /* Selection du mode alpha 25 x 80 couleur avec le bit "Enable Blink
       Attribute" a zero afin d'obtenir 16 couleurs pour l'attribut
       "Background Color".*/
    outp(MODE_SELECT_REGISTER, ALPHA_16);
    /* Selection de la couleur de contour de l'ecran */
    outp(COLOR_SELECT_REGISTER, BORDER_COLOR);
    /* Initialisation de image_alpha[] avec chr$(220) */
    pimage_alpha = image_alpha;
    i = 3840;
    while (i) {
       *pimage_alpha++ = 220;
       i--;
    }
    /* Transfert de image_alpha[] vers la memoire de la carte video */
    poke(VIDEO_PAGE1_SEGMENT, 0, image_alpha, 3840);
    /* Ecriture de la ligne 25 */
    pligne25 = ligne25_attribut;
    ligne25_offset = LIGNE25_BASE + 1;
    i = 80;
    while (i) {
       poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, pligne25++, 1);
       ligne25_offset += 2;
       i--;
    }
    pligne25 = ligne25_commande;
    ligne25_offset = LIGNE25_BASE;
    i = 80;
    while (i) {
       poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, pligne25++, 1);
       ligne25_offset += 2;
       i--;
    }
    /* Initialisation de la table palette_attribut[] */
    for (j=0; j<16; j++) {
       for (i=0; i<16; i++) {
          palette_attribut[16 * j + i] =
             palette_couleur[i] + palette_couleur[j] * 16;
       }
    }
    return;
}

/*
```

```
*/
/***************************************************************************

                        ecriture_mode(mode)                      p-(12)

****************************************************************************

Cette routine va ecrire le mode actuel dans la zone de mode de la ligne 25.

***************************************************************************/

ecriture_mode(mode)

int mode;

{
   static char m1[16] = {" Mode Direct:"};
   static char m2[16] = {" Mode Sauve:"};
   static char m3[16] = {" Mode Rappel:"};
   static char m4[16] = {" Mode Edisque:"};
   static char m5[16] = {" Mode Ldisque:"};
   char *pmode;
   int i, ligne25_offset;

   switch (mode) {
      case 1: pmode = m1; break;
      case 2: pmode = m2; break;
      case 3: pmode = m3; break;
      case 4: pmode = m4; break;
      case 5: pmode = m5; break;
      default: break;
   }

   ligne25_offset = LIGNE25_BASE + 86;
   i = 15;
   while (i) {
      poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, pmode++, 1);
      ligne25_offset += 2;
      i--;
   }
   return;
}
/*
```

```
*/
/*****************************************************************************

                        ecriture_numero_image(numero)

*****************************************************************************

Cette routine va ecrire le numero de l'image sauvee dans la zone de reponse
de la ligne 25.

*****************************************************************************/

ecriture_numero_image(numero)

int numero;

{
   int i, ligne25_offset;
   static char numero_ecran[5];
   char *pecran;

   pecran = numero_ecran;
   i = 5;
   while (i) {
      *pecran++ = 0;
      i--;
   }

   stci_d(numero_ecran, numero, 5);

   pecran = numero_ecran;
   ligne25_offset = LIGNE25_BASE + 154;
   i = 3;

   while(i) {
      poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, pecran++, 1);
      ligne25_offset += 2;
      i--;
   }
   return;
}

/*
```

```
*/
/*****************************************************************************

                        ecriture_question(question)

*****************************************************************************

Cette routine va ecrire une question dans une zone de la ligne 25.

*****************************************************************************/

ecriture_question(question)

int question;

{
   static char q1[20] = {"Image(00-29) ?"};
   static char q2[20] = {"Fichier(000-999) ?"};
   static char q3[20] = {"Commandes:  + - A"};
   char *pquestion;
   int i, ligne25_offset;

   switch (question) {
      case 1: pquestion = q1; break;
      case 2: pquestion = q2; break;
      case 3: pquestion = q3; break;
      default: break;
   }

   ligne25_offset = LIGNE25_BASE + 116;
   i = 18;
   while (i) {
      poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, pquestion++, 1);
      ligne25_offset += 2;
      i--;
   }
   return;
}
/*
```

```
*/
/*****************************************************************************

                          incremente(compteur)

******************************************************************************

Cette routine incremente le compteur de profil en mode rappel.

*****************************************************************************/

incremente(compteur)

int compteur;

{
   compteur++;
   if (compteur > 29) {
      compteur = 000;
   }
   return(compteur);
}
/*
```

```
*/
/*****************************************************************************

                        lancement_acquisition()                    P-16

*****************************************************************************

Cette routine active l'acquisition d'une image provenant de la camera infra-
rouge par la memoire double-acces de la carte AI.  La carte opere en mode
rafraichissement sur demande.  On verifie le statut pour une fin de cycle
normal.  Autrement, on imprime un message d'erreur avec la valeur du statut
et on termine l'execution du programme.  La variable time_out empeche le
programme d'attendre indefiniment pour un statut valide.

*****************************************************************************/

lancement_acquisition()
{
    char statut, commande;
    int time_out, delay;

    commande = MODE_RFD;
    poke(CONTROL_SEGMENT, R0, &commande, 1);
    commande = RAZ;
    poke(CONTROL_SEGMENT, R1, &commande, 1);
    commande = RCT;
    poke(CONTROL_SEGMENT, R1, &commande, 1);
    commande = CRF;
    poke(CONTROL_SEGMENT, R1, &commande, 1);

    statut = 0;
    time_out = 0;
    while (statut != FIN_CYCLE && time_out < 5000) {
        peek(CONTROL_SEGMENT, R0, &statut, 1);
        time_out++;
    }
    if (statut != FIN_CYCLE) {
        printf("\nErreur sur le cycle d'acquisition: statut =%4d", statut);
        printf("\ntime_out = %5d", time_out);
        retour();
        exit(-1);
    }
        return;
}


/*
```

```
*/
/*****************************************************************************

                        memorise_image(numero_image)                    p-17

*****************************************************************************

Cette routine fait le transfert du tampon image_brute[] vers une zone de
memoire protegee i.e. non-occupee par le programme PSEUDO.EXE.  Cette zone
sert de tampon d'images en vue d'un eventuel transfert vers l'unite de
disque.  Une image occupe 3840 octets, on dispose d'une zone sure de 128 K
(de 0X50000 a 0X70000).  Ceci implique donc une capacite de 30 images au
maximum (#0 a #29).  L'argument de la fonction est donc le numero d'image.
La variable segment_image depend de l'argument numero_image.  Le facteur
multiplicatif est egal a 240, soit 3840 octets divise par 16 (segmentation
du 8088).

*****************************************************************************/

memorise_image(numero_image)

int numero_image;

{
   extern char image_brute[];
   int segment_image;

   segment_image = TAMPON_SEGMENT + numero_image * 240;
   if (numero_image > 29) {
      printf("\nErreur: numero d'image trop grand");
      return(-1);
   }
   poke(segment_image, 0, image_brute, 3840);
   return(segment_image);
}

/*
```

```
*/
/*****************************************************************************

                                mode_direct()

*****************************************************************************

Cette routine permet d'afficher en temps reel les images de la camera infra-
rouge.

*****************************************************************************/

mode_direct()
{
   ecriture_mode(1);
   while (!kbhit()) {
      transfert_image();
   }
   vidange_dialogue();
   return;
}
/*
```

```
*/
/*****************************************************************************

                          mode_ecriture_disque()                  P-(19)

******************************************************************************

Cette routine va ecrire un fichier binaire sur disque.  Ce fichier contient
les donnees de 30 images situees dans la zone de memoire protegee.

*****************************************************************************/

mode_ecriture_disque()
{
   extern char image_brute[];
   extern int _fmode;
   FILE *pfichier;
   char suffixe[5];
   static char nom_fichier[] = {"B:ESSAI.XXX"};
   char *psuffixe, *pnom, *pbrute;
   int i, grosseur_bloc, nombre_bloc, bloc_transfere, numero;

   ecriture_mode(4);
   numero = -1;       /* numero de fichier non valide */
   while ((numero = demande_numero_fichier()) > 999 || numero < 0);
   stci_d(suffixe, numero, 5);
   psuffixe = suffixe;
   pnom = nom_fichier + 8;
   i = 3;
   while (i) {
      *pnom++ = *psuffixe++;
      i--;
   }
   _fmode = 0X8000;
   grosseur_bloc = 48;
   nombre_bloc = 80;
   pbrute = image_brute;
   if ((pfichier = fopen(nom_fichier, "w") ) == NULL) {
      printf("\nImpossible d'ouvrir le fichier %15s", nom_fichier);
      retour();
      exit(-1);
   }
   vidange_dialogue();
   ecriture_mode(4);
   for (i=0; i<30; i++) {
      ecriture_numero_image(i);
      rappel_image(i);
      bloc_transfere =
        fwrite(pbrute, grosseur_bloc, nombre_bloc, pfichier);
      if (bloc_transfere < nombre_bloc) {
         printf("\nErreur de transaction au disque\n");
         retour();
         exit(-1);
      }
   }
   if (fclose(pfichier) == -1) {
      printf("\nErreur sur la fermeture du fichier");
      retour();
      exit(-1);
   }
```

```
    _fmode = 0X0000;
    vidange_dialogue();
    return;
}
/*
```

```
*/
/*****************************************************************************

                        mode_lecture_disque()                    p-(21)

*****************************************************************************

Cette routine va lire un fichier binaire sur disque pour aller le placer
dans la zone de memoire protegee.  Le fichier a une capacite de 30 images.

*****************************************************************************/

mode_lecture_disque()
{
   extern char image_brute[];
   extern int _fmode;
   FILE *pfichier;
   char suffixe[5];
   static char nom_fichier[] = {"B:ESSAI.XXX"};
   char *psuffixe, *pnom, *pbrute;
   int i, grosseur_bloc, nombre_bloc, bloc_transfere, numero;

   ecriture_mode(5);
   numero = -1;       /* numero de fichier non valide */
   while ((numero = demande_numero_fichier()) > 999 || numero < 0);
   stci_d(suffixe, numero, 3);
   psuffixe = suffixe;
   pnom = nom_fichier + 8;
   i = 3;
   while (i) {
      *pnom++ = *psuffixe++;
      i--;
   }
   _fmode = 0X8000;
   grosseur_bloc = 48;
   nombre_bloc = 80;
   pbrute = image_brute;
   if ((pfichier = fopen(nom_fichier, "r") ) == NULL) {
      printf("\nImpossible d'ouvrir le fichier %15s", nom_fichier);
      retour();
      exit(-1);
   }
   vidange_dialogue();
   ecriture_mode(5);
   for (i=0; i<30; i++) {
      ecriture_numero_image(i);
      bloc_transfere =
        fread(pbrute, grosseur_bloc, nombre_bloc, pfichier);
      if (bloc_transfere < nombre_bloc) {
         printf("\nErreur de transaction au disque\n");
         retour();
         exit(-1);
      }
      memorise_image(i);
   }
   if (fclose(pfichier) == -1) {
      printf("\nErreur sur la fermeture du fichier");
      retour();
      exit(-1);
   }
```

```
    _fmode = 0X0000;
    vidange_dialogue();
    return;
}
/*
```

```
*/
/*************************************************************************

                        mode_rappel()                    D-(23)

**************************************************************************

Cette routine permet a l'operateur de visionner la sequence de 30 images
dans la zone de memoire protegee.

*************************************************************************/

mode_rappel()

{
   int compteur, retour_menu_principal;

   compteur = 0;
   retour_menu_principal = 0;
   ecriture_mode(3);
   ecriture_question(3);
   ecriture_numero_image(compteur);
   rappel_image(compteur);
   affiche_image();
   while (!retour_menu_principal) {
      if (kbhit()) {
         switch (getch()) {
            case '+': compteur = incremente(compteur); break;
            case '-': compteur = decremente(compteur); break;
            case 'A': case 'a': compteur = avance(compteur); break;
            default : retour_menu_principal = 1; break;
         }
         ecriture_numero_image(compteur);
         rappel_image(compteur);
         affiche_image();
      }
   }
   vidange_dialogue();
   return;
}

/*
```

```
*/

/*****************************************************************************

                                mode_sauve()

*****************************************************************************

Cette routine permet a l'operateur d'initier le transfert en temps reel de
30 images consecutives dans la zone de memoire protegee.

*****************************************************************************/

mode_sauve()

{
   int i;

   ecriture_mode(2);
   for (i=0; i<30; i++) {
      ecriture_numero_image(i);
      transfert_image();
      memorise_image(i);
      if (kbhit()) {
         i = 29;
      }
   }
   vidange_dialogue();
   return;
}
/*
```

```
*/

/*****************************************************************************

                        rappel_image(numero_image)

*****************************************************************************

Cette routine fait le transfert d'une image de la zone de memoire protegee
vers le tampon image_brute.  Cette fonction est la contre-partie de la
fonction memorise_image().

*****************************************************************************/

rappel_image(numero_image)

int numero_image;

{
   extern char image_brute[];
   int segment_image;

   segment_image = TAMPON_SEGMENT + numero_image * 240;
   if (numero_image > 33) {
      printf("\nErreur: numero d'image trop grand");
      return(-1);
   }
   peek(segment_image, 0, image_brute, 3840);
   return(segment_image);
}
/*
```

```
*/
/***************************************************************************

                              retour()

****************************************************************************

Cette routine effectue le retour a l'environnement du DOS.  Le controleur
video 6845 est reinitialise a la page zero dans la memoire d'affichage.

***************************************************************************/

retour()
{
   outp(INDEX_REGISTER, 12);
   outp(DATA_REGISTER, 0X00);
   outp(INDEX_REGISTER, 13);
   outp(DATA_REGISTER, 0X00);
}

/*
```

P-(26)

```
*/
/************************************************************************

                         transfert_image()

*************************************************************************

Cette routine fait le transfert d'une image entre la camera infra-rouge et
l'affichage video.

************************************************************************/

transfert_image()

{
   lancement_acquisition();
/* acquisition_image(); */
   acquisition_80_colonne();
   affiche_image();
   return;
}
/*
```

P-(27)

```
*/
/*****************************************************************************

                                vidange_dialogue()

*****************************************************************************

Cette routine remet a zero les caracteres de la zone de dialogue, soit la
zone de mode, la zone de question et la zone de reponse.

*****************************************************************************/

vidange_dialogue()
{
   int i, ligne25_offset;
   char valeur;

   valeur = 0;
   ligne25_offset = LIGNE25_BASE + 86;
   i = 37;
   while (i) {
      poke(VIDEO_PAGE1_SEGMENT, ligne25_offset, &valeur, 1);
      ligne25_offset += 2;
      i--;
   }
   return;
}

/*
```

P-(28)

ANNEXE B

Logiciel SUIVI.C

```
/* 5 mai 1986
*****************************************************************************

                                SUIVI.C

*****************************************************************************

    Version 2.0
(c) Tous droits reserves,
    Yves Chartier, Institut de Soudage du Canada, 1986

*****************************************************************************
*/

#include <stdio.h>

#define ALPHA_16                    9
#define BORDER_COLOR                0
#define COLOR_SELECT_REGISTER    0X3D9
#define CONTROL_SEGMENT         0X8800
#define CRF                         1
#define DATA_REGISTER            0X3D5
#define FIN_CYCLE                   3
#define INDEX_REGISTER           0X3D4
#define INT_MASK_REGISTER        0X021
#define MODE_RFD                    0
#define MODE_SELECT_REGISTER     0X3D8
#define RCT                         8
#define R0                          0
#define R1                          1
#define RAM_SEGMENT             0X8C00
#define RAZ                         6
#define STATUS_REGISTER          0X3DA
#define SUP_MASK                 0XF0
#define TAMPON_SEGMENT          0X5000
#define TIMER_INT_OFF            0X0AD
#define TIMER_INT_ON             0X0AC
#define VERTICAL_MASK               8
#define VIDEO_DISABLE               0
#define VIDEO_PAGE0_SEGMENT     0XB800

/*
```

```
*/


char image_brute[3840];
char image_alpha[4000];
char palette_attribut[255];
char *line_pointer_array[24];
int profil[48];
int _fmode = 0X0000;
int colonne_zero = 105;
int colonne_profil = 127;
int taux_memorisation = 1;
int premier_senseur = 16;
int position_centre;
int derivee[17];
int offset_x = 30;
int statut_profil = 1;
int commande_filtree;

main()
{
   int commande_sortie;

   depart();

   commande_sortie = 0;
   while (!commande_sortie) {
      if (kbhit()) {
         switch (getch()) {
            case 'd': case 'D':mode_direct(); break;
            case 'm': case 'M':mode_memorise(); break;
            case 'r': case 'R':mode_rappel(); break;
            case 's': case 'S':mode_suivi(); break;
            case 'e': case 'E':mode_ecriture_disque(); break;
            case 'l': case 'L':mode_lecture_disque(); break;
            case 'f': case 'F':commande_sortie = 1; break;
            case 'z': case 'Z':mode_colonne_zero(); break;
            case 'p': case 'P':mode_colonne_profil(); break;
            case 't': case 'T':mode_taux_memorisation(); break;
            case 'b': case 'B':mode_ligne_basse(); break;
            case 'o': case 'O':mode_offset(); break;
            default: break;
         }
      }
   }
}
/*
```

```
*/
/*****************************************************************************

                        acquisition_profil()

*****************************************************************************

Cette routine fait l'acquisition d'un profil relatif en faisant la soustrac-
tion entre le profil_absolu[] et le profil_zero[].  Ce resultat est
transfere au tampon externe profil[].  Les variables externes colonne_zero
et colonne_profil determinent les colonnes choisies dans l'image.

*****************************************************************************/

acquisition_profil()

{
   extern int colonne_zero, colonne_profil, profil[];
   static char profil_zero[48], profil_absolu[48];
   int i, *prelatif;
   char *pzero, *pabsolu;

   peek(RAM_SEGMENT, colonne_zero * 64, profil_zero, 48);
   peek(RAM_SEGMENT, colonne_profil * 64, profil_absolu, 48);

   pzero = profil_zero;
   pabsolu = profil_absolu;
   prelatif = profil;
   i = 48;
   while (i) {
      *prelatif++ = *pabsolu++ - *pzero++;
      i--;
   }

   return;
}
/*
```

```
*/
/*****************************************************************************

                              affiche_derivee()

*****************************************************************************

Cette routine affiche 17 points du tampon derivee[].

*****************************************************************************/

affiche_derivee()
{
   extern int derivee[];
   static int courbe[17], courbe_precedente[17];
   int *pcourbe, *pcourbe_precedente, *pderivee;
   int i, offset0, offset_dollar, statut0, statut1;
   char valeur0, valeur_dollar;

   /* Transfert de courbe[] a courbe_precedente[] */
   pcourbe = courbe;
   pcourbe_precedente = courbe_precedente;
   for (i=0; i<17; i++) {
      *pcourbe_precedente++ = *pcourbe++;
   }

   /* Calcul de la nouvelle courbe */
   pcourbe = courbe;
   pderivee = derivee + 16;  /* en decroissant */
   for (i=0; i<17; i++) {
      *pcourbe = ((*pderivee-- >> 3) << 1) + 50;
      if (*pcourbe < 0) {
         *pcourbe = 0;
      }
      if (*pcourbe > 100) {
         *pcourbe = 100;
      }
      pcourbe++;
   }

   /* Mise a jour de l'affichage video */
/* outp(INT_MASK_REGISTER, TIMER_INT_OFF);  desactive le timer */
   pcourbe = courbe;
   pcourbe_precedente = courbe_precedente;
   valeur0 = 0;
   valeur_dollar = '$';
   statut0 = 0;
   statut1 = 0;
         /*  Detection du video vertical enable: transition de 0 a 1 */
   while (!(!statut0 && statut1)) {
      statut0 = inp(STATUS_REGISTER) & VERTICAL_MASK;
      statut1 = inp(STATUS_REGISTER) & VERTICAL_MASK;
   }
   for (i=0; i<9; i++) {
      offset0 = (i + 4) * 160 + *pcourbe_precedente++;  /* efface */
      poke(VIDEO_PAGE0_SEGMENT, offset0, &valeur0, 1);
      offset_dollar = (i + 4) * 160 + *pcourbe++;       /* affiche */
      poke(VIDEO_PAGE0_SEGMENT, offset_dollar, &valeur_dollar, 1);
   }
   statut0 = 0;
```

```
    statut1 = 0;
          /*  Detection du video vertical enable: transition de 0 a 1 */
    while (!(!statut0 && statut1)) {
       statut0 = inp(STATUS_REGISTER) & VERTICAL_MASK;
       statut1 = inp(STATUS_REGISTER) & VERTICAL_MASK;
    }
    for (i=9; i<17; i++) {
       offset0 = (i + 4) * 160 + *pcourbe_precedente++;  /* efface */
       poke(VIDEO_PAGE0_SEGMENT, offset0, &valeur0, 1);
       offset_dollar = (i + 4) * 160 + *pcourbe++;       /* affiche */
       poke(VIDEO_PAGE0_SEGMENT, offset_dollar, &valeur_dollar, 1);
    }
/* outp(INT_MASK_REGISTER, TIMER_INT_ON);  reactive le timer */
    return;
}

/*
```

```
*/
/*****************************************************************************

                              affiche_profil()

*****************************************************************************

Cette routine affiche 21 points du tampon profil[] a partir de la variable
externe premier_senseur.

*****************************************************************************/

affiche_profil()
{
   extern int profil[], premier_senseur;
   static int courbe[21], courbe_precedente[21];
   int *pcourbe, *pcourbe_precedente, *pointeur_profil;
   int i, offset0, offset_etoile, statut0, statut1;
   char valeur0, valeur_etoile;

   /* Transfert de courbe[] a courbe_precedente[] */
   pcourbe = courbe;
   pcourbe_precedente = courbe_precedente;
   for (i=0; i<21; i++) {
      *pcourbe_precedente++ = *pcourbe++;
   }

   /* Calcul de la nouvelle courbe */
   pcourbe = courbe;
   pointeur_profil = profil + 20 + premier_senseur;  /* en decroissant */
   for (i=0; i<21; i++) {
      *pcourbe = (*pointeur_profil-- >> 1) << 1;
/*    *pcourbe = ((*pointeur_profil-- - 140) >> 1) << 1;
   ***** vieille version pour essais 000-009                    */
      if (*pcourbe < 0) {
         *pcourbe = 0;
      }
      if (*pcourbe > 114) {
         *pcourbe = 114;
      }
      pcourbe++;
   }

   /* Mise a jour de l'affichage video */
/* outp(INT_MASK_REGISTER, TIMER_INT_OFF);  desactive le timer */
   pcourbe = courbe;
   pcourbe_precedente = courbe_precedente;
   valeur0 = 0;
   valeur_etoile = '*';
   statut0 = 0;
   statut1 = 0;
         /*  Detection du video vertical enable: transition de 0 a 1 */
   while (!(!statut0 && statut1)) {
      statut0 = inp(STATUS_REGISTER) & VERTICAL_MASK;
      statut1 = inp(STATUS_REGISTER) & VERTICAL_MASK;
   }
   for (i=0; i<11; i++) {
      offset0 = (i + 2) * 160 + *pcourbe_precedente++;  /* efface */
      poke(VIDEO_PAGE0_SEGMENT, offset0, &valeur0, 1);
      offset_etoile = (i + 2) * 160 + *pcourbe++;       /* affiche */
```

```
      poke(VIDEO_PAGE0_SEGMENT, offset_etoile, &valeur_etoile, 1);
   }
   statut0 = 0;
   statut1 = 0;
         /*  Detection du video vertical enable: transition de 0 a 1 */
   while (!(!statut0 && statut1)) {
      statut0 = inp(STATUS_REGISTER) & VERTICAL_MASK;
      statut1 = inp(STATUS_REGISTER) & VERTICAL_MASK;
   }
   for (i=11; i<21; i++) {
      offset0 = (i + 2) * 160 + *pcourbe_precedente++;  /* efface */
      poke(VIDEO_PAGE0_SEGMENT, offset0, &valeur0, 1);
      offset_etoile = (i + 2) * 160 + *pcourbe++;       /* affiche */
      poke(VIDEO_PAGE0_SEGMENT, offset_etoile, &valeur_etoile, 1);
   }
/* outp(INT_MASK_REGISTER, TIMER_INT_ON);  reactive le timer */
   return;
}
/*
```

S-①

```
*/
/*****************************************************************************

                              analyse_profil()

*****************************************************************************

Cette routine analyse les donnees du tampon profil[] pour en extraire la
position du centre du joint ainsi que la hauteur de la torche.  De plus,
elle donne une acceptation ou un rejet du profil analyse selon certains
criteres.

*****************************************************************************/

analyse_profil()

{
   calcul_derivee();
/* affiche_derivee();                pour fin d'analyse preliminaire */
   calcul_centre_hauteur();
   calcul_commande_centre();
   return;
}

/*
```

```
*/
/*****************************************************************************

                              avance(compteur)

*****************************************************************************

Cette routine fait defiler successivement les profils en mode rappel.

*****************************************************************************/

avance(compteur)

int compteur;

{
   int i;

   while (!kbhit()) {
      compteur++;
      if (compteur > 999) {
         compteur = 000;
      }
      ecriture_numero(24, compteur);
      rappel_profil(compteur);
      analyse_profil();
      affiche_profil();
   }
   return(compteur);
}
/*
```

```
*/
/*****************************************************************************

                        calcul_derivee()

*****************************************************************************

Cette routine calcul la derivee du profil affiche.   Le calcul est fait de
la facon suivante:

   Soit 5 points consecutifs, la derivee normalisee devient alors:

               derivee = (p5 - p1) + (p4 -p2 )/4

   Pour ne pas perdre de resolution  on va plutot prendre:

               derivee = 4 * (p5 - p1) + (p4 - p2)

La routine de calcul utilise 21 points du tampon profil[] pour generer les
17 points de la derivee..

*****************************************************************************/

calcul_derivee()

{
   extern int profil[], derivee[], premier_senseur;
   int i, *pointeur_profil, *pderivee, *p1, *p2, *p4, *p5;

   p1 = profil + premier_senseur ;
   p2 = profil + premier_senseur + 1;
   p4 = profil + premier_senseur + 3;
   p5 = profil + premier_senseur + 4;
   pderivee = derivee;
   i = 17;

   while (i) {
      *pderivee++ = ((*p5++ - *p1++) << 2) + *p4++ - *p2++;
      i--;
   }
   return;
}

/*
```

```
*/
/*****************************************************************************

                        decremente(compteur)

*****************************************************************************

Cette routine decremente le compteur de profil en mode rappel.

*****************************************************************************/

decremente(compteur)

int compteur;

{
   compteur--;
   if (compteur < 000) {
      compteur = 999;
   }
   return(compteur);
}

/*
```

```
*/
/*****************************************************************************

                        demande_numero(nombre_digit)

*****************************************************************************

Cette routine va chercher au clavier un numero qui contient un nombre de
chiffres variable.  La zone de message de la ligne 24 affiche les entrees
validee provenant du clavier.

*****************************************************************************/


demande_numero(nombre_digit)

int nombre_digit;

{
   char entree_clavier[6];
   char *pclavier;
   char digit, valeur;
   int i, offset, numero;

   valeur = 0;
   offset = 3968;          /* colonne 64 & ligne 24 */
   pclavier = entree_clavier;

   i = 4;
   while (i) {
      poke (VIDEO_PAGE0_SEGMENT, offset, &valeur, 1);
      offset += 2;
      i--;
      }
   offset = 3968;         /* colonne 64 & ligne 24 */
   while (nombre_digit) {
      if (kbhit()) {
         digit = getch();
         if (digit >= '0' && digit <= '9') {
            *pclavier = digit;
            poke (VIDEO_PAGE0_SEGMENT, offset, pclavier, 1);
            nombre_digit--;
            offset += 2;
            pclavier++;
         }
         if (digit == '\b') {
            return(-1);     /* Valeur de retour non valide a etre testee  */
         }                  /* par l'appelant pour pouvoir recommencer la */
      }                     /* la routine.                                */
   }
   *pclavier++ = '\0';
   stcd_i(entree_clavier, &numero);
   return(numero);
}

/*
```

```
*/
/*****************************************************************************

                                depart()

*****************************************************************************

Cette routine de depart du programme SUIVI.C procede a l'initialisation de
la carte d'affichage video en mode alphanumerique/couleur.  Cette routine
genere aussi l'ecran de travail.

*****************************************************************************/

depart()
{
   extern char image_alpha[];
   extern int colonne_zero, colonne_profil, taux_memorisation;
   static int registre_6845[16] =
      { 0X71, 0X50, 0X5A, 0X0A, 0X1F, 0X06, 0X19, 0X1C,
        0X02, 0X07, 0X06, 0X07, 0X00, 0X00, 0X20, 0X00
      };
   static char ligne0_graphic[] =
      {"0   10   20   30   40   50   60   70   80   90   100  110 ");
   static char ligne1_graphic[] =
      {"¦----¦----¦----¦----¦----¦----¦----¦----¦----¦----¦----¦--");
   static char colonne58_graphic[] =
      {"-    -    -    -    -"};
   static char colonne59_graphic[] =
      {"¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦"};
   static char colonne60_graphic[] =
      {"+    +    -    -"};
   static char colonne61_graphic[] =
      {"1    5    0    5    1"};
   static char colonne62_graphic[] =
      {"0                   0"};
   static char ligne_attribut[80];
   int i, j, *pregistre;
   int offset;
   char *pimage_alpha, *pattribut;
   char  *pgraphic58, *pgraphic59, *pgraphic60, *pgraphic61, *pgraphic62;
   char  *p0graphic, *p1graphic, *p23graphic, *p24graphic;

   /* Desactivation du signal video */
   outp(MODE_SELECT_REGISTER, VIDEO_DISABLE);
   /* Initialisation du 6845 pour le mode 25 x 80 couleur */
   pregistre = registre_6845;
   for (i=0; i<16; i++) {
      outp(INDEX_REGISTER, i);
      outp(DATA_REGISTER, *pregistre++);
   }
   /* Selection de la couleur de contour de l'ecran */
   outp(COLOR_SELECT_REGISTER, BORDER_COLOR);
   /* Initialisation de image_alpha[] avec chr$(0) */
   pimage_alpha = image_alpha;
   i = 4000;
   while (i) {
      *pimage_alpha++ = 0;
      i--;
   }
   /* Transfert de image_alpha[] vers la memoire de la carte video */
```

```
poke(VIDEO_PAGE0_SEGMENT, 0, image_alpha, 4000);
/* Ecriture des attributs */
pattribut = ligne_attribut;
for (i=0; i<63 ; i++) {
   *pattribut++ = 31;
}
for (i=63; i<80 ; i++) {
   *pattribut++ = 160;
}
for (j=0; j<80; j++) {
   pattribut = ligne_attribut;
   offset = (j * 160) + 1;
   i = 80;
   while (i) {
      poke(VIDEO_PAGE0_SEGMENT, offset, pattribut++, 1);
      offset += 2;
      i--;
   }
}
/* Ecriture du graphique */
p0graphic  = ligne0_graphic;     /* ligne0  */
p1graphic  = ligne1_graphic;     /* ligne1  */
p23graphic = ligne1_graphic;     /* ligne23 */
p24graphic = ligne0_graphic;     /* ligne24 */
offset = 0;                       /* colonne 0 & ligne 0 */
i = 58;
while (i) {
   poke(VIDEO_PAGE0_SEGMENT, offset, p0graphic++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 160, p1graphic++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 3680, p23graphic++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 3840, p24graphic++, 1);
   offset += 2;
   i--;
}

pgraphic58 = colonne58_graphic;     /* colonne 58 */
pgraphic59 = colonne59_graphic;     /* colonne 59 */
pgraphic60 = colonne60_graphic;     /* colonne 60 */
pgraphic61 = colonne61_graphic;     /* colonne 61 */
pgraphic62 = colonne62_graphic;     /* colonne 62 */
offset = 436;                       /* colonne 58 & ligne 2 */
i = 21;
while (i) {
   poke(VIDEO_PAGE0_SEGMENT, offset, pgraphic58++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 2, pgraphic59++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 4, pgraphic60++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 6, pgraphic61++, 1);
   poke(VIDEO_PAGE0_SEGMENT, offset + 8, pgraphic62++, 1);
   offset += 160;
   i--;
}

for (i=0; i<23; i++) {
   ecriture_message(i, i);
}
for (i=10; i<21; i++) {
   ecriture_attribut_message(i,95);
}
ecriture_attribut_message(21, 224);
ecriture_attribut_message(22, 224);
```

```
    ecriture_attribut_message(23, 224);
    ecriture_attribut_message(24, 207);
    ecriture_numero(13, offset_x);
    ecriture_numero(17, taux_memorisation);
    ecriture_numero(15, colonne_zero);
    ecriture_numero(16, colonne_profil);
    ecriture_numero(20, premier_senseur);
    /* Selection du mode alpha 25 x 80 couleur avec le bit "Enable Blink
       Attribute" a zero afin d'obtenir 16 couleurs pour l'attribut
       "Background Color".*/
    outp(MODE_SELECT_REGISTER, ALPHA_16);

    return;

}
/*
```

```
*/
/*****************************************************************************

                    ecriture_attribut_message(ligne, attribut)

*****************************************************************************

Cette routine assigne un attribut de couleur pour une ligne de la zone de
messages.

*****************************************************************************/

ecriture_attribut_message(ligne, attribut)

int ligne, attribut;

{
    int i, offset;
    char valeur;

    /* Ecriture du message */
    valeur = attribut;
    offset = (ligne * 160) + 127;  /* colonne # 63 */
    i = 17;
    while (i) {
       poke(VIDEO_PAGE0_SEGMENT, offset, &valeur, 1);
       offset += 2;
       i--;
    }
    return;
}

/*
```

```
*/
/*****************************************************************************

                    ecriture_message(ligne, numero_message)

*****************************************************************************

Cette routine va ecrire un message sur une ligne de la zone de messages.

*****************************************************************************/

ecriture_message(ligne, numero_message)

int ligne, numero_message;
{
   static char *message[] = {
          "                    ",     /*  0 */
          "----- MODES -----",     /*  1 */
          " Direct           ",     /*  2 */
          " Memorise/Rappel  ",     /*  3 */
          " Suivi            ",     /*  4 */
          " Edisque/Ldisque  ",     /*  5 */
          " Fin              ",     /*  6 */
          " col. Zero/Profil",     /*  7 */
          " Taux de mem.     ",     /*  8 */
          " ligne Bas/Offset",     /*  9 */
          "-- ETAT ACTUEL --",     /* 10 */
          " Sommet_0 =       ",     /* 11 */
          " Sommet_1 =       ",     /* 12 */
          " Offset   =       ",     /* 13 */
          " Position =       ",     /* 14 */
          " Czero    =       ",     /* 15 */
          " Cprofil  =       ",     /* 16 */
          " Taux mem.=       ",     /* 17 */
          " Statut_P =       ",     /* 18 */
          " Fichier  =       ",     /* 19 */
          " ligne Bas=       ",     /* 20 */
          "                  ",     /* 21 */
          "------------------",     /* 22 */
          "  +  -  Avance    ",     /* 23 */
          " Zero(000-255)?   ",     /* 24 */
          " Profil(000-255)?",     /* 25 */
          " Suivi(000-999)?  ",     /* 26 */
          " Taux(1, 2, 3)?   ",     /* 27 */
          " Profil no.       ",     /* 28 */
          " Ligne (00-27)?   ",     /* 29 */
          " Offset(00-60)?   "      /* 30 */
   };
   int i, offset;
   char *pmessage;

   offset = (ligne * 160) + 126;  /* colonne # 63 */
   pmessage = message[numero_message];
   i = 17;
   while (i) {
      poke(VIDEO_PAGE0_SEGMENT, offset, pmessage++, 1);
      offset += 2;
      i--;
   }
   return;
```

}
/*

S-⑲

```
*/
/*****************************************************************************

                              ecriture_mode(mode)

*****************************************************************************

Cette routine va ecrire le mode actuel dans la zone de mode de la ligne 21.

*****************************************************************************/

ecriture_mode(mode)

int mode;

{
   static char *message[] = {
          " Mode Direct:     ",     /* 0 */
          " Mode Memorise:   ",     /* 1 */
          " Mode Rappel:     ",     /* 2 */
          " Mode Suivi:      ",     /* 3 */
          " Mode Edisque:    ",     /* 4 */
          " Mode Ldisque:    ",     /* 5 */
          " Mode Zero:       ",     /* 6 */
          " Mode Profil:     ",     /* 7 */
          " Mode Taux:       ",     /* 8 */
          " Mode ligne Bas:  ",     /* 9 */
          " Mode Offset:     "      /* 10 */
   };

   int i, offset;
   char *pmessage;

   /* Ecriture du message */
   offset = 3486;  /* colonne # 63 & ligne # 21 */
   pmessage = message[mode];
   i = 17;
   while (i) {
      poke(VIDEO_PAGE0_SEGMENT, offset, pmessage++, 1);
      offset += 2;
      i--;
   }
   return;
}
/*
```

```
*/
/*****************************************************************************

                        ecriture_numero(ligne, numero)

*****************************************************************************

Cette routine va ecrire un nombre entier de 3 chiffres dans la zone de
parametres d'une ligne.

*****************************************************************************/

ecriture_numero(ligne, numero)

int ligne, numero;

{
   int i;
   static char message[5];
   char *pmessage;

   pmessage = message;
   i = 5;
   while (i) {
      *pmessage++ = 0;
      i--;
   }

   stci_d(message, numero, 5);
   ecriture_parametre(ligne, message);
   return;
}
/*
```

```
*/
/*****************************************************************************

                    ecriture_parametre(ligne, pmessage)

*****************************************************************************

Cette routine va ecrire la valeur d'un parametre sur une ligne de la zone
de parametres.  Cette valeur est deja une chaine de caracteres.

*****************************************************************************/

ecriture_parametre(ligne, pmessage)

int ligne;
char *pmessage;

{
    int i, offset;

    /* Ecriture du message */
    offset = (ligne * 160) + 150;  /* colonne # 75 */
    i = 5;
    while (i) {
       poke(VIDEO_PAGE0_SEGMENT, offset, pmessage++, 1);
       offset += 2;
       i--;
    }
    return;
}

/*
```

```
*/
/*****************************************************************************

                        incremente(compteur)

*****************************************************************************

Cette routine incremente le compteur de profil en mode rappel.

*****************************************************************************/

incremente(compteur)

int compteur;

{
   compteur++;
   if (compteur > 999) {
      compteur = 000;
   }
   return(compteur);
}

/*
```

```
*/

/*****************************************************************************

                        lancement_acquisition()

*****************************************************************************

Cette routine active l'acquisition d'une image provenant de la camera infra-
rouge par la memoire double-acces de la carte AI.  La carte opere en mode
rafraichissement sur demande.  On verifie le statut pour une fin de cycle
normal.  Autrement, on imprime un message d'erreur avec la valeur du statut
et on termine l'execution du programme.  La variable time_out empeche le
programme d'attendre indefiniment pour un statut valide.

*****************************************************************************/

lancement_acquisition()
{
    char statut, commande;
    int time_out, delay;

    commande = MODE_RFD;
    poke(CONTROL_SEGMENT, R0, &commande, 1);
    commande = RAZ;
    poke(CONTROL_SEGMENT, R1, &commande, 1);
    commande = RCT;
    poke(CONTROL_SEGMENT, R1, &commande, 1);
    commande = CRF;
    poke(CONTROL_SEGMENT, R1, &commande, 1);

    statut = 0;
    time_out = 0;
    while (statut != FIN_CYCLE && time_out < 5000) {
        peek(CONTROL_SEGMENT, R0, &statut, 1);
        time_out++;
    }
    if (statut != FIN_CYCLE) {
        printf("\nErreur sur le cycle d'acquisition: statut =%4d", statut);
        printf("\ntime_out = %5d", time_out);
        exit(-1);
    }
        return;
}


/*
```

```
*/
/*****************************************************************************

                        memorise_profil(numero_profil)

*****************************************************************************

Cette routine fait le transfert du tampon profil[] vers une zone de
memoire protegee i.e. non-occupee par le programme SUIVI.EXE.  Cette zone
sert de tampon de profils en vue d'un eventuel transfert vers l'unite de
disque.  Une profil occupe 96 octets, on dispose d'une zone sure de 128 K
(de 0X50000 a 0X70000).  Ceci implique donc une capacite d'au moins 1000
profils.  L'argument de la fonction est donc le numero de profil.La variable
segment_image depend de l'argument numero_profil.  Le facteur multiplicatif
est egal a 6, soit 96 octets divise par 16 (segmentation du 8088).

*****************************************************************************/

memorise_profil(numero_profil)

int numero_profil;

{
   extern int profil[];
   char *pointeur_profil;
   int segment_profil;

   pointeur_profil = (char *) profil;
   segment_profil = TAMPON_SEGMENT + numero_profil * 6;
   if (numero_profil > 999) {
      printf("\nErreur: numero de profil trop grand");
      return(-1);
   }
   poke(segment_profil, 0, pointeur_profil, 96);
   return(segment_profil);
}
/*
```

```
*/
```



```
/*****************************************************************************

                        mode_colonne_profil()

*****************************************************************************

Cette routine permet de changer la colonne d'acquisition du profil.

*****************************************************************************/

mode_colonne_profil()

{
   extern int colonne_profil;
   int numero;

   ecriture_mode(7);
   ecriture_message(23,25);
   while ((numero = demande_numero(3)) > 255 || numero < 0);
   colonne_profil = numero;
   ecriture_numero(16, numero);
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                        mode_colonne_zero()

*****************************************************************************

Cette routine permet de changer la colonne de reference du zero.

*****************************************************************************/

mode_colonne_zero()

{
   extern int colonne_zero;
   int numero;

   ecriture_mode(6);
   ecriture_message(23,24);
   while ((numero = demande_numero(3)) > 255 || numero < 0);
   colonne_zero = numero;
   ecriture_numero(15, numero);
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                                mode_direct()

*****************************************************************************

Cette routine permet d'afficher en temps reel le profil selectionne.

*****************************************************************************/

mode_direct()
{
   ecriture_mode(0);
   while (!kbhit()) {
      transfert_profil();
      analyse_profil();
      affiche_profil();
   }
   vidange_message(21);
   return;
}
/*
```

```
*/
/*****************************************************************************

                        mode_ecriture_disque()

*****************************************************************************

Cette routine va ecrire un fichier binaire sur disque.  Ce fichier contient
les donnees de 1000 profils situes dans la zone de memoire protegee.

*****************************************************************************/

mode_ecriture_disque()
{
   extern int profil[];
   extern int _fmode;
   FILE *pfichier;
   char suffixe[5];
   static char nom_fichier[] = {"B:SUIVI.XXX"};
   char *psuffixe, *pnom, *pointeur_profil;
   int i, grosseur_bloc, nombre_bloc, bloc_transfere, numero;

   ecriture_mode(4);
   ecriture_message(23, 26);
   numero = -1;       /* numero de fichier non valide */
   while ((numero = demande_numero(3)) > 999 || numero < 0);
   ecriture_numero(19, numero);
   vidange_message(23);
   stci_d(suffixe, numero, 5);
   psuffixe = suffixe;
   pnom = nom_fichier + 8;
   i = 3;
   while (i) {
      *pnom++ = *psuffixe++;
      i--;
   }
   _fmode = 0X8000;
   grosseur_bloc = 96;
   nombre_bloc = 1;
   pointeur_profil = (char *) profil;
   if ((pfichier = fopen(nom_fichier, "w") ) == NULL) {
      printf("\nImpossible d'ouvrir le fichier %15s", nom_fichier);
      exit(-1);
   }
   ecriture_message(24, 28);
   for (i=0; i<1000; i++) {
      ecriture_numero(24, i);
      rappel_profil(i);
      bloc_transfere =
        fwrite(pointeur_profil, grosseur_bloc, nombre_bloc, pfichier);
      if (bloc_transfere < nombre_bloc) {
         printf("\nErreur de transaction au disque\n");
         exit(-1);
      }
   }
   if (fclose(pfichier) == -1) {
      printf("\nErreur sur la fermeture du fichier\n");
      exit(-1);
   }
   _fmode = 0X0000;
```

```
    vidange_message(21);
    vidange_message(24);
    return;
}
/*
```

```
*/
/*****************************************************************************

                              mode_ligne_basse()

*****************************************************************************

Cette routine permet de changer le premier senseur de la fenetre d'affichage

*****************************************************************************/

mode_ligne_basse()

{
   extern int premier_senseur;
   int numero;

   ecriture_mode(9);
   ecriture_message(23,29);
   while ((numero = demande_numero(2)) > 27 || numero < 0);
   premier_senseur = numero;
   ecriture_numero(20, numero);
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                         mode_lecture_disque()

******************************************************************************

Cette routine va lire un fichier binaire sur disque pour aller le placer
dans la zone de memoire protegee.  Le fichier a une capacite total de
1000 profils.

*****************************************************************************/

mode_lecture_disque()
{
   extern int profil[];
   extern int _fmode;
   FILE *pfichier;
   char suffixe[5];
   static char nom_fichier[] = {"B:SUIVI.XXX"};
   char *psuffixe, *pnom, *pointeur_profil;
   int i, grosseur_bloc, nombre_bloc, bloc_transfere, numero;

   ecriture_mode(5);
   ecriture_message(23, 26);
   numero = -1;        /* numero de fichier non valide */
   while ((numero = demande_numero(3)) > 999 || numero < 0);
   ecriture_numero(19, numero);
   vidange_message(23);
   stci_d(suffixe, numero, 5);
   psuffixe = suffixe;
   pnom = nom_fichier + 8;
   i = 3;
   while (i) {
      *pnom++ = *psuffixe++;
      i--;
   }
   _fmode = 0X8000;
   grosseur_bloc = 96;
   nombre_bloc = 1;
   pointeur_profil = (char *) profil;
   if ((pfichier = fopen(nom_fichier, "r") ) == NULL) {
      printf("\nImpossible d'ouvrir le fichier %15s", nom_fichier);
      exit(-1);
   }
   ecriture_message(24, 28);
   for (i=0; i<1000; i++) {
      ecriture_numero(24, i);
      bloc_transfere =
        fread(pointeur_profil, grosseur_bloc, nombre_bloc, pfichier);
      if (bloc_transfere < nombre_bloc) {
         printf("\nErreur de transaction au disque\n");
         exit(-1);
      }
      memorise_profil(i);
   }
   if (fclose(pfichier) == -1) {
      printf("\nErreur sur la fermeture du fichier\n");
      exit(-1);
   }
```

```
    _fmode = 0X0000;
    vidange_message(21);
    vidange_message(24);
    return;
}
/*
```

```
*/
/*****************************************************************************

                              mode_memorise()

*****************************************************************************

Cette routine permet a l'operateur d'initier le transfert en temps reel de
1000 profils consecutifs dans la zone de memoire protegee.

*****************************************************************************/

mode_memorise()

{
   extern int taux_memorisation;
   int i, j, jmax;

   ecriture_mode(1);
   ecriture_message(24, 28);
   jmax = taux_memorisation;
   for (i=0; i<1000; i++) {
      for (j=0; j<jmax; j++) {
         ecriture_numero(24, i);
         transfert_profil();
         memorise_profil(i);
         analyse_profil();
         affiche_profil();
      }
      if (kbhit()) {
         i = 999;
      }
   }
   vidange_message(21);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                                mode_offset()

*****************************************************************************

Cette routine permet d'ajuster le decalage sur la commande de l'axe x.

*****************************************************************************/

mode_offset()

{
   extern int offset_x;
   int numero;

   ecriture_mode(10);
   ecriture_message(23,30);
   while ((numero = demande_numero(2)) > 60 || numero < 0);
   offset_x = numero;
   ecriture_numero(13, numero);
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                              mode_rappel()

*****************************************************************************

Cette routine permet a l'operateur de visionner la sequence de 1000 profils
dans la zone de memoire protegee.

*****************************************************************************/

mode_rappel()

{
   int compteur, retour_menu_principal;

   compteur = 0;
   retour_menu_principal = 0;
   ecriture_mode(2);
   ecriture_message(23, 23);
   ecriture_message(24, 28);
   ecriture_numero(24, compteur);
   rappel_profil(compteur);
   analyse_profil();
   affiche_profil(compteur);
   while (!retour_menu_principal) {
      if (kbhit()) {
         switch (getch()) {
            case '+': compteur = incremente(compteur); break;
            case '-': compteur = decremente(compteur); break;
            case 'A': case 'a': compteur = avance(compteur); break;
            default : retour_menu_principal = 1; break;
         }
         ecriture_numero(24, compteur);
         rappel_profil(compteur);
         analyse_profil();
         affiche_profil();
      }
   }
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}
/*
```

```
/*****************************************************************************

                                mode_suivi()

*****************************************************************************

Cette routine fait le suivi tout en faisant le transfert en temps reel de
1000 profils consecutifs dans la zone de memoire protegee.

*****************************************************************************/

mode_suivi()

{
   extern int taux_memorisation;
   int i, j, jmax, flag;

   ecriture_mode(3);
   ecriture_message(24, 28);
   initialisation_cy512();
   jmax = taux_memorisation;
   flag = 0;
   for (i=0; i<1000; i++) {
      for (j=0; j<jmax; j++) {
         ecriture_numero(24, i);
         transfert_profil();
         memorise_profil(i);
         analyse_profil();
/*       affiche_profil();          pas d'affichage pour accelerer  */
         lancement_commande();
      }
      if (kbhit()) {
         i = 999;
         flag = 1;
      }
   }
                              /* continue a suivre sans enregistrement */
   if (!flag) {
      for (i=0; i<10000; i++) {
         transfert_profil();
         analyse_profil();
         lancement_commande();
         if (kbhit()) {
            i = 9999;
         }
      }
   }
   reset_cy512();
   vidange_message(21);
   vidange_message(24);
   return;
}
/*
```

```
*/
/*****************************************************************************

                        mode_taux_memorisation()

*****************************************************************************

Cette routine permet de changer le taux de memorisation des profils en mode
memorise ou en mode suivi.

*****************************************************************************/

mode_taux_memorisation()

{
   extern int taux_memorisation;
   int numero;

   ecriture_mode(8);
   ecriture_message(23,27);
   while ((numero = demande_numero(1)) > 3 || numero < 0);
   taux_memorisation = numero;
   ecriture_numero(17, numero);
   vidange_message(21);
   vidange_message(23);
   vidange_message(24);
   return;
}

/*
```

```
*/
/*****************************************************************************

                        rappel_profil(numero_profil)                    5- (57)

*****************************************************************************

Cette routine fait le transfert d'un profil de la zone de memoire protegee
vers le tampon profil[]e.  Cette fonction est la contre-partie de la
fonction memorise_profil().

*****************************************************************************/

rappel_profil(numero_profil)

int numero_profil;

{
   extern int profil[];
   int segment_profil;
   char *pointeur_profil;

   pointeur_profil = (char *) profil;
   segment_profil = TAMPON_SEGMENT + numero_profil * 6;
   if (numero_profil > 999) {
      printf("\nErreur: numero de profil trop grand");
      return(-1);
   }
   peek(segment_profil, 0, pointeur_profil, 96);
   return(segment_profil);
}
/*
```

```
*/

/****************************************************************************

                    transfert_profil()

*****************************************************************************

Cette routine fait le transfert d'un profil entre la camera infra-rouge et
l'affichage video.

****************************************************************************/

transfert_profil()

{
   lancement_acquisition();
   acquisition_profil();
   return;
}

/*
```

5 (39)

# ANNEXE C

Logiciel SUITE.C

```
/* 5 mai 1986 */
/*****************************************************************************

                                SUITE.C

                    ( deuxieme partie de SUIVI.C )

*****************************************************************************

     Version 2.0
(c)  Tous droits reserves,
     Yves Chartier, Institut de Soudage du Canada, 1986

*****************************************************************************/

#include <stdio.h>


#define CS_ON             0X12      /* mode binaire CS = ON */
#define CS_OFF            0X92      /* mode binaire CS = OFF */
#define CONTROLE_X       0X300
#define CONTROLE_Y       0X301
#define STATUT_X         0X300
#define STATUT_Y         0X301
#define PORT_DATA        0X302
#define CY512_PRET        0XA0      /* bit5 = 1 et bit7 = 1 */
#define CY512_RECU        0X00      /* bit7 = 0 */
#define PRET_MASK         0XA0      /* bit5 et bit 7 */
#define RECU_MASK         0X80      /* bit7 */
#define RESET_CY512       0X02      /* reset */

/*
```

```
*/
/*****************************************************************************

                        calcul_centre_hauteur()

*****************************************************************************

Cette routine calcule la position du centre en vue du positionnement selon
l'axe x ainsi que la hauteur relative pour le positionnement selon l'axe y.
Il est a noter que l'echelle de l'axe x comporte 10 divisions par senseur.
L'offset de -80 correspond au centre des 17 points de la derivee.

*****************************************************************************/

calcul_centre_hauteur()

{
    extern int derivee[];
    extern int position_centre, statut_profil;
    int i, s, *p1, *p2, *psommet;
    static int sommet[25];

    p1 = derivee;
    p2 = derivee + 1;
    psommet = sommet;
    s = 0;
    for (i=0; i<16; i++) {
        if (*p1>0 && *p2<=0) {
            *psommet++ = -80 + 10 * i + (*p1 * 10)/(*p1 -*p2);
            s++;
        }
        p1++;
        p2++;
    }
    if (s!=2) {
        statut_profil = 0;
        ecriture_numero(18, statut_profil);
        vidange_parametre(14);
        vidange_parametre(11);
        vidange_parametre(12);
    }
    else {
        statut_profil = 1;
        ecriture_numero(18, statut_profil);
        position_centre = sommet[0] + ((sommet[1] - sommet[0]) >> 1);
        ecriture_numero(14, position_centre);
        ecriture_numero(11, sommet[0]);
        ecriture_numero(12, sommet[1]);
    }
    return;
}
/*
```

```
*/
/*****************************************************************************

                        calcul_commande_centre()

*****************************************************************************

Cette routine fait le calcul de la commande envoyee a l'axe x pour aller
au centre du joint.

La position du centre est d'abord corrigee a l'aide d'un decalage ajustable
offset_y,.  On fait ensuite la conversion pour la commande au moteur, le
facteur est le suivant:

                F = (7500 um / 200 div) / (25 um / step) = - 1.5

Finalement, la commande est filtree a l'aide d'un passe-bas du premier ordre
selon la configuration suivante:

                OUT(t) = K * IN(t)  +  (1-K) * OUT(t-1)

     commande_filtree(t) = .25 * commande(t) + .75 commande_filtree(t-1)

*****************************************************************************/

calcul_commande_centre()

{
   extern int position_centre, offset_x, commande_filtree, statut_profil;
   int i, position_corrigee, commande;

   position_corrigee = position_centre -30 + offset_x;

                       /* statut_profil == 0  ==>  pas de commande */
   if ('statut_profil) {
      position_corrigee = 0;
   }

   commande = -((position_corrigee << 1) + position_corrigee >> 1);
   commande_filtree =
      ((commande_filtree << 1) + commande_filtree + commande) >> 2;
   return;

}
/*
```

```
*/
/*****************************************************************************

                        initialisation_cy512()

*****************************************************************************

Cette routine procede a l'initialisation du controleur de moteur CY512.

*****************************************************************************/

initialisation_cy512()

{
   static int x_vecteur[24] =
      { 'I', 0, 'R', 1, 254, 'S', 1, 25, 'F', 1, 75,
        'N', 2, 1, 0, '+', 0, 'E', 0, 'G', 0, '0', 0, 'Q'    };
   static int y_vecteur[24] =
      { 'I', 0, 'R', 1, 215, 'S', 1, 25, 'F', 1, 1,
        'N', 2, 1, 0, '+', 0, 'E', 0, 'G', 0, '0', 0, 'Q'    };
   int i, delai, *pvecteur;

   outp(CONTROLE_Y, RESET_CY512);     /* reset hardware */
   delai = 100;
   while (delai--);
   outp(CONTROLE_Y, CS_OFF);          /* mode binaire */
   pvecteur = y_vecteur;             /* transfert du y_vecteur */
   i = 24;
   while (i) {
      y_commande(*pvecteur++);
      i--;
   }

   outp(CONTROLE_X, RESET_CY512);     /* reset hardware */
   delai = 100;
   while (delai--);
   outp(CONTROLE_X, CS_OFF);          /* mode binaire */
   pvecteur = x_vecteur;             /* transfert du x_vecteur */
   i = 24;
   while (i) {
      x_commande(*pvecteur++);
      i--;
   }

   return;
}
/*
```

```
*/
/*****************************************************************************

                        lancement_commande()

*****************************************************************************

Cette routine active la commande des moteurs a partir des valeurs calculees.

*****************************************************************************/

lancement_commande()

{
   extern int commande_filtree;

   x_deplacement(commande_filtree);
   return;

}
/*
```

```
*/
/*****************************************************************************

                                reset_cy512()

*****************************************************************************

Cette routine remet a zero le controleur de moteur CY512.

*****************************************************************************/

reset_cy512()

{
   outp(CONTROLE_X, RESET_CY512);
   outp(CONTROLE_Y, RESET_CY512);
   return;
}

/*
```

```
*/
/*****************************************************************************

                        x_commande(commande)

*****************************************************************************

Cette routine envoie un caractere de commande de l'axe X au controleur
de moteur CY512 en respectant un protocole de communication.

*****************************************************************************/

x_commande(commande)

int commande;                       /* la commande doit etre < 256 */

{
   int statut, time_out;

   statut = -1;                                                 /* verifie si pret */
   time_out = 0;
   while (statut != CY512_PRET  && time_out < 10000) {
      statut = inp(STATUT_X) & PRET_MASK;
      time_out++;
   }
   if (statut != CY512_PRET) {
      printf("\nErreur X_CY512_PRET : statut =%4d", statut);
      printf("\ntime_out = %5d", time_out);
      exit(-1);
   }
   outp(PORT_DATA, commande);          /* place la commande */
   statut = -1;                         /* verifie si bien recu */
   time_out = 0;
   outp(CONTROLE_X, CS_ON);            /* active le chip select */
   while (statut != CY512_RECU  && time_out < 10000) {
      statut = inp(STATUT_X) & RECU_MASK;
      time_out++;
   }
   if (statut != CY512_RECU) {
      printf("\nErreur X_CY512_RECU : statut =%4d", statut);
      printf("\ntime_out = %5d", time_out);
      exit(-1);
   }
   outp(CONTROLE_X, CS_OFF);           /* desactive le chip select */

   return;
}
/*
```

```
*/
/*****************************************************************************

                        x_deplacement(nombre_pas)

*****************************************************************************

Cette routine envoie la sequence de caracteres necessaire au controleur de
moteur de l'axe x afin d'executer un deplacement.

*****************************************************************************/

x_deplacement(nombre_pas)

int nombre_pas;

{
   static int vecteur_commande[8] =  { 'N', 2, 1, 0, '+', 0, 'D', 0 };
   int i, valeur_absolue;
   int *pvecteur;

   if (nombre_pas < 0) {
      vecteur_commande[4] = '-';
      valeur_absolue = - nombre_pas;
   }
   else {
      vecteur_commande[4] = '+';
      valeur_absolue = nombre_pas;
   }
   vecteur_commande[2] = valeur_absolue & 0XFF;
   vecteur_commande[3] = (valeur_absolue >> 8) & 0X7F;

   pvecteur = vecteur_commande;
   i = 8;
   while (i) {
      x_commande(*pvecteur++);
      i--;
   }

   return;
}
/*
```

```
*/
/*****************************************************************************

                        y_commande(commande)

*****************************************************************************

Cette routine envoie un caractere de commande de l'axe Y au controleur de
moteur CY512 en respectant un protocole de communication.

*****************************************************************************/

y_commande(commande)

int commande;                       /* la commande doit etre < 256 */

{
   int statut, time_out;

   statut = -1;                                              /* verifie si pret */
   time_out = 0;
   while (statut != CY512_PRET  && time_out < 10000) {
      statut = inp(STATUT_Y) & PRET_MASK;
      time_out++;
   }
   if (statut != CY512_PRET) {
      printf("\nErreur Y_CY512_PRET: statut =%4d", statut);
      printf("\ntime_out = %5d", time_out);
      exit(-1);
   }
   outp(PORT_DATA, commande);          /* place la commande */
   statut = -1;
   time_out = 0;
   outp(CONTROLE_Y, CS_ON);            /* active le chip select */
                                       /* verifie si bien recu */
   while (statut != CY512_RECU  && time_out < 10000) {
      statut = inp(STATUT_Y) & RECU_MASK;
      time_out++;
   }
   if (statut != CY512_RECU) {
      printf("\nErreur Y_CY512_RECU: statut =%4d", statut);
      printf("\ntime_out = %5d", time_out);
      exit(-1);
   }
   outp(CONTROLE_Y, CS_OFF);           /* desactive le chip select */

   return;
}
/*
```

```
*/
/*****************************************************************************

                        LISTE DES FONCTIONS

*****************************************************************************

calcul_centre_hauteur()
calcul_commande_centre()
initialisation_cy512()
lancement_commande()
reset_cy512()
x_commande(commande)
x_deplacement(nombre_pas)
y_commande(commande)

*****************************************************************************
/*
```